ESTIMATION OF EVOKED FIELDS USING A TIME-SEQUENCED ADAPTIVE FILTER WITH THE MODIFIED P-VECTOR ALGORITHM

THESIS

Jeffery A. Kepley
Captain, USAF

AFIT/EN/GE90D

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/EN/GE90D

ESTIMATION OF EVOKED FIELDS USING A TIME-SEQUENCED ADAPTIVE FILTER WITH THE MODIFIED P-VECTOR ALGORITHM

THESIS

Jeffery A. Kepley
Captain, USAF

AFIT/EN/GE90D

AFIT/EN/GE90D

ESTIMATION OF EVOKED FIELDS USING A TIME-SEQUENCED ADAPTIVE FILTER WITH THE MODIFIED P-VECTOR ALGORITHM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Jeffery A. Kepley, B.S.

Captain, USAF

December, 1990

Preface

The purpose of this study was to apply adaptive filter signal processing techniques to the processing of visually evoked fields and visually evoked potentials which are signals generated by the human brain. The signal-to-noise ratio of these signals is extremely low due to the presence of strong background noise. Through the use of a time sequenced adaptive filter, an estimate of the evoked fields was generated using the evoked potentials. While the material presented does not require any special background, a general knowledge of digital signal processing and adaptive filter techniques is useful.

As a student looking for the reasons why, I want to thank my advisor Dr Robert Williams. His patience, advice, and insight were greatly needed and appreciated. I would also like to thank those on the thesis committee; Lt Col D. Meer, Lt Col D. Norman, and Dr M. Kabrisky.

I am most indebted and thankful to my best friend and wife, Kristi. Her support and encouragement made this all possible. Also, a warm thanks to my two wonderful boys, Joshua and Jacob. Their smiles, laughter, and hugs kept the "dad" in me alive. Finally, a prayer of thanksgiving to the Heavenly Father for his blessings and support.

Jeffery A. Kepley

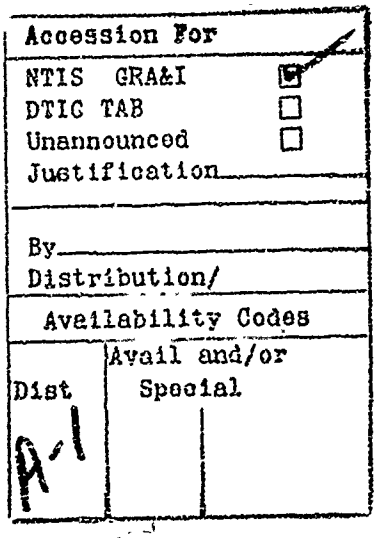

Table of Contents

List of Figures

List of Tables

Abstract

A time sequenced adaptive filter is developed to estimate visually evoked fields (EF) using visually evoked potentials (EP). These non-stationary signals are buried in strong background noise. The two types of noise are magnetoencephalogram (MEG) and electroencephalogram (EEG). The filter implementation is based on the Ferrara Time Sequenced Adaptive Filter (TSAF) using the Least-Mean-Square (LMS) algorithm and the Williams modified P-vector algorithm (mPa). This essentially results in two filters, the $TSAF_{LMS}$ and the $TSAF_{mPa}$ respectively. A two stage filter structure is used in which the first stage removes the time-varying mean of the input signals. This allows the second stage to process zero-mean signals which increases the convergence speed of the filter.

The theory for the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters is overviewed with the input signals to the filters modelled as the sum of three uncorrelated components: average signal response, signal jitter, and noise. The signal model is verified based on a statistical analysis of simulated EP data files. The software implementation is then shown to be error free. Using simulated EF and EP signals buried in correlated noise, the $TSAF_{mPa}$ is shown to out perform the $TSAF_{LMS}$ for the specific case presented. The $TSAF_{mPa}$ is unique in that the filter uses an estimate of the cross correlation statistics of the pre-stimulus noise in the filter update equation. This allows the $TSAF_{mPA}$ to remove the biasing affects of cross correlated noise which the $TSAF_{LMS}$ can not do. The noise statistics, contained in the P_n vector, are assumed to be time-invariant which is a fundamental assumption used in this research.

Prior to filtering human data files, a statistical analysis performed on human pre-stimulus noise reveals correlation between the MEG and EEG noise components for the specific data files used. The human data is then filtered using the $TSAF_{LMS}$ and $TSAF_{mPa}$ followed by a comparison of the output signals.

ESTIMATION OF EVOKED FIELDS USING A TIME-SEQUENCED ADAPTIVE FILTER WITH THE MODIFIED P-VECTOR ALGORITHM

I. Introduction

1.1 Background

The human brain generates two signals in response to visual and/or auditory stimuli, the electrical evoked potential (EP) and the magnetic evoked field (EF). These signals are difficult to measure as they consist of weak signals buried in strong background noise. The two types of background noise are electroencephalogram (EEG) and magnetoencephalogram (MEG) (15:5). In addition to the low signal to noise ratio (SNR), the EP and EF signals are nonstationary or time-varying such that classic Wiener filters are inadequate for processing these signals (13:1).

The EF and EP signals are used to locate the source of the response(s) to stimuli in the human brain. Location of the source is considered to be the most important piece of information in biomagnetic research because the location helps researchers understand how the brain processes information (8:9-10). The Armstrong Aerospace Medical Research Laboratory (AAMRL) conducts research in the application of EP and EF to source location as it applies to pilot work load assessment and developing techniques for optimally integrating pilot and cockpit environments. One proposed approach for assessing the suitability of cockpit designs is to quantify the brain response of pilots to various workload scenarios for different cockpit configurations. Quantifying the brain's response requires access to the information component of measured electro-magnetic brain responses which is then used to localize the response region. AAMRL collects EF and EP signals from human subjects as part of

this on going study and desires to recover the human EF signals from low SNR data signals collected to date. The problem encountered is the nonstationarity and low SNR of the EF and EP signals. While this thesis is directed towards processing EF signals, the literature on the signal processing of EF is limited compared to that on the EP signal. Therefore, the reader is now presented a brief history of EP signal processing keeping in mind that both EP and EF are nonstationary and corrupted with strong background noise.

In the past, ensemble averaging has been used to enhance the SNR of the EP signal. The major drawback to this is that some of the important information content of the signal is averaged out in the process (15:32). Westerkamp pointed out that Wiener filters were used to process EP will little or no improvement over averaging. This is attributed to the nonstationarity of the EP signal and the fact that Wiener based filters, in general, require a priori knowledge of the signal statistics which are assumed stationary (9:13). Use of adaptive filters has also been attempted with results similar to those obtained from using Wiener filters. Although a prior knowledge of the signal statistics is not required, the adaptive filter will converge to an approximation of the Wiener solution which assumes stationary signal statistics (13:5). It has been shown that the Least-Mean-Square (LMS) adaptive filter can track signals with slowly changing statistics by increasing the adaptation gain. The trade off is that increasing the gain adds adaptation noise to the solution (11) (12:34). Given the similarity of the EF and EP signals, one can expect similar results from processing EF signals. With an obvious need for a filter which can process nonstationary signals, Ferrara developed the Time Sequenced Adaptive Filter (TSAF) (2).

The Ferrara TSAF is an extension of the LMS filter and is designed to process a certain class of nonstationary signals which have statistics that repeat in time or are cyclostationary (2:22). The TSAF requires two input signals as does the LMS filter and thus requires two separate sensors to collect the signals. Researchers have suggested that the EF and EP signals contain correlated information which

possibly makes the EF and EP signals good candidates for the TSAF (8:16-17) (15). Going a step further, one might desire to use only one sensor and still process the nonstationary signal adaptively. This is the idea behind the Williams modified P-Vector algorithm (mPa) which essentially estimates the statistics of the pre-stimulus noise and then removes the estimate from the weight update. This assumes the noise has stationary statistics with the input signal allowed to be nonstationary. The mPa requires "pre-stimulus" data which is the ongoing background noise collected before the stimulus is applied (12:94-97). Two filter implementations are investigated in this thesis. The TSAF using the LMS algorithm and the TSAF using the Williams mPa. These are denoted as the $TSAF_{LMS}$ and $TSAF_{mPa}$ respectively.

1.2 Problem Statement

The objective of this thesis is to estimate EF signals using EP signals in a direct application of the Ferrara TSAF and estimate EF signals by incorporating the Williams Modified P-Vector Algorithm into the TSAF.

1.3 Approach

The plan of attack for this thesis includes a literature search, a statistical analysis of simulated EP signals, a $TSAF_{LMS}$ and $TSAF_{mPa}$ filter implementation, testing of the software, and an estimation of human EF signals. The following is a brief overivew of the remaining chapters:

- Chapter II. This chapter presents the necessary background information to understand the source, model, and statistical characteristics of the EP and EF signals. In addition, the theory behind the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters is overviewed.
- Chapter III. This chapter discusses the $TSAF_{LMS}$ and $TSAF_{mPa}$ filter implementations in software and presents the test that were used to verify the

integrity of the filters. This chapter also develops the concept of using EP to estimate EF by testing both the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters with simulated EP and simulated EF signals.

- Chapter IV. With the software implementation verified, this chapter presents the results from estimating human EF using three different filter configurations.

  1. Two Sensor $TSAF_{LMS}$.
  2. Two Sensor $TSAF_{mPa}$.
  3. Single Sensor $TSAF_{mPA}$.

- Chapter V. This chapter presents the conclusions from this research effort and recommendations for future research.

1.4 Expected Results

The direct results from this thesis include a working $TSAF_{LMS}$ and $TSAF_{mPa}$ program and a report on the results from using the filters on the human EF and human EP signals. Other expected results which are presented in Chapter 3 include the following:

- Test data which verifies that the $TSAF_{LMS}$ and $TSAF_{mPa}$ programs are essentially error free.
- Statistical characteristics of the human EEG and MEG noise.
- Comparison of the $TSAF_{LMS}$ and $TSAF_{mPa}$ performance in terms of the mean-square error.

1.5 Hardware Requirements

The implementation of the adaptive filter requires an AT style personal computer with a 512K Ram Disk, 640K of base memory, and a single floppy drive. No additional hardware is required.

1.6 Software Requirements

The filter program required Borland Turbo Pascal 5.0 compiler, editor and debugger. Statistical analysis required Mathcad and Borland Turbo Basic. The Basic programs were used to read and write unformatted data files and perform some recursive calculations which are not easy to implement in Mathcad.

II. Background

2.1 Introduction

The topic of this chapter is the application of adaptive filter signal processing to estimating the magnetic evoked fields (EF) from the human brain using human evoked potentials (EP). The topic is developed by presenting information obtained from a literature search of the applicable material. In addition, a statistical analysis is presented on simulated EP signals. The key terms associated with this topic are biomagnetism, magnetic evoked field, adaptive filter, and time sequenced adaptive filter (TSAF).

2.1.1 Definition of Key Terms. Biomagnetism is a broad area of research that deals with the collection and analysis of signals emanated by the human body. EF and EP signals are a category of nonstationary biomagnetic signals that are produced by the brain in response to visual an ,'or auditory stimuli (15:32). Adaptive filters are a class of filters which are self adjusting in response to their input signals . In general, the adaptive filters adjust to optimize some performance criteria (10:5). The TSAF is an extension of the broad category of adaptive filters that is considered for use with the EF signal (2:22). These terms are discussed further in the following sections.

2.1.2 Scope of the Research Topic and Data Base. The area of biomagnetism is fairly well documented and is a significant area of ongoing research. The scope of the literature search for this thesis will be limited to EF and EP signals generated by visual stimuli. This will reduce the general topic of biomagnetism to a manageable area. The data base of information is contained in conference reports, technical journals, and text books.

Adaptive filters are widely used for signal processing and control systems. A significant amount of literature is available on adaptive filters. However, the filter of interest for this paper is a closed-loop adaptive filter which uses the least-mean-square (LMS) algorithm. Using the key words LMS and closed-loop adaptive filter greatly reduces the broad topic of adaptive filters. The main source of information for the LMS adaptive filter is text books and technical journals. The source for the TSAF is limited to class lectures, technical articles, and Ferrara's PhD dissertation (2) (3) (12).

2.1.3 Method of Treatment and Organization. The following sections address the EF and EP signals first, followed by a short discussion of adaptive filter theory. The EF and EP literature will be overviewed to introduce the source and source model of the EF and EP signals. In addition, a brief description of how the signals are collected is presented as well. This is followed by an analysis of the statistical characteristics of simulated EP signals. The section on the adaptive filter will highlight major points and results of classical LMS adaptive filter theory. The theory presented on the LMS adaptive filter will then be incorporated into the discussion of the Ferrara TSAF and the Williams modified P-vector algorithm (mPa).

2.2 *Definition and Analysis of the EF and EP Signals*

The purpose of this section is to describe how the EF and EP signals are generated, modeled for analysis, and collected by AAMRL. This is followed by a discussion on the statistical characteristics of simulated EP signals.

2.2.1 Source and Model of the EF and EP Signals. The EF and EP signals are generated from the brain in response to visual and/or auditory stimuli. This thesis is only concerned with the generation due to visual stimuli. When a nerve cell is stimulated beyond some threshold, the ionic equilibrium of the cell is changed. This change in the ionic state of the nerve cell generates an electric current which propagates along the nerve to connecting nerves or tissues (14:143). For the eye, the propagation path is along the optical nerve to the visual cortex (14:360). In order to study and analyze this phenomena, the process is modeled as a simple dipole with the nerve path as a thin wire from which a magnetic field is generated (14:404-405).

It is generally known that a magnetic field is generated by the flow of current along a wire and the magnetic field forms a circular pattern around the wire. In addition, the direction of the magnetic field is determined by the direction the current is traveling along the wire (14:20). Thus, the EF signal is the magnetic field generated by the flow of current along the nerve path and provides information as to the orientation and, more importantly, the location of the source (8:9-10). The EP signal is also generated from the dipole model and is the potential produced from the stimulus which drives the current. Given the source and the nature of the EF and EP signals, it is obvious that the signals are of very low power and susceptible to noise corruption, therefore, the collection of these signals requires special equipment.

2.2.2 Signal Collection. AAMRL uses a Superconducting Quantum Interference Device (SQUID) to collect the EF signals which is a non-invasive means of recording magnetic fields generated by the brain. The SQUID uses a superconducting current loop which is extremely sensitive to changes in magnetic flux densities

which are associated with the EF signal (15:8). The first reliable SQUID was introduced in 1969 and since has been improved to allow measurements of brain activity without special shielding to eliminate noise from overhead lighting or other sources of urban noise (8:5). The EP signal is collected by electrodes attached to the subject head which record the potential generated from the stimulus (15:25).

The data for this thesis was collected from a human subject who was looking at a checkerboard pattern which was repeatedly stimulated by a burst of light. The EF and EP data is time synchronized to the burst of light and is formatted into a two column matrix and stored as an ASCII file on a floppy disk (15:18-20).

2.2.3 Statistical Analysis of Simulated EP Signals. This section provides a statistical analysis of simulated EP data. The purpose is to determine specific statistical characteristics of the signal which include the following: stationarity, autocorrelation, mean, and ensemble average. The analysis is based on the following signal vector model:

$$X_j = M_j + Q_j + N_j \tag{2.1}$$

M_j is the average signal response or ensemble average vector at trial j, Q_j is the deviation vector of the signal about the average response and is called the "jitter", and N_j is the human pre-stimulus noise vector. This is the signal model originally used by Williams and provides additional insight into the performance of the adaptive filter in terms of signal bias and correlation of the individual signal components (13:40). This will become evident in the development of the adaptive filter theory which follows. Two data files were used for the analysis. The first is SDAT which contains a time indexed signal composed of the average response signal and the jitter vectors, M_j and Q_j respectively. The second file called DDAT contains the SDAT signal with human EEG noise, N_j, added to the signal or $M_j + Q_j + N_j$. Both data files contained 100 vectors and each vector was composed of 50 discrete data points. All the individual signal components were extracted and saved in separate ASCII

files. The following sections discuss the analysis performed for M_j, Q_j, and N_j with a definition of the statistical formulas provided in Appendix A.

2.2.3.1 Average Response Signal, M_j. The first signal component analyzed was the M_j vector which was obtained by ensemble averaging the SDAT data file. Ensemble averaging effectively removes all the zero mean signal components, including the jitter, from the signal leaving the average response signal. M_j is deterministic which means it does not change with each data vector and the j subscript could be dropped (M). Therefore, any variations in X_j from trial to trial are due to the zero mean jitter and/or noise components. Figure 2.1 shows the plot of the ensemble average of SDAT which produces the signal M_j. The plot shows that the time sequenced mean or ensemble average of X_j is time varying and thus, by definition, M_j is nonstationary.

2.2.3.2 Signal Jitter, Q_j. The signal jitter or Q_j vector was obtained by subtracting the M_j vector from all 100 data vectors contained in SDAT or

$$Q_j = SDAT_j - M_j \tag{2.2}$$

The subscript on M_j could be dropped as it is assumed to be the same for each data vector. The first step was to determine what the Q_j component of the signal looked like. Figure 2.2 shows three of the Q data vectors plotted as a function of the time index k (compare with Figure 2.1). The signal varied significantly among each of the separate data vectors as can be seen from the plot. However, the signal appears to vary symmetrically about the time axis which would indicate Q_j has a zero mean or $E[q_k] = 0$. The mean was calculated and the result indicates that Q_j is zero mean (see Figure 2.3). Next, using all 100 vectors, the time sequenced ensemble mean-square or variance of Q_j was calculated and plotted as a function of the time index k (Figure 2.4). It is obvious from the plot that the signal's second order statistics are

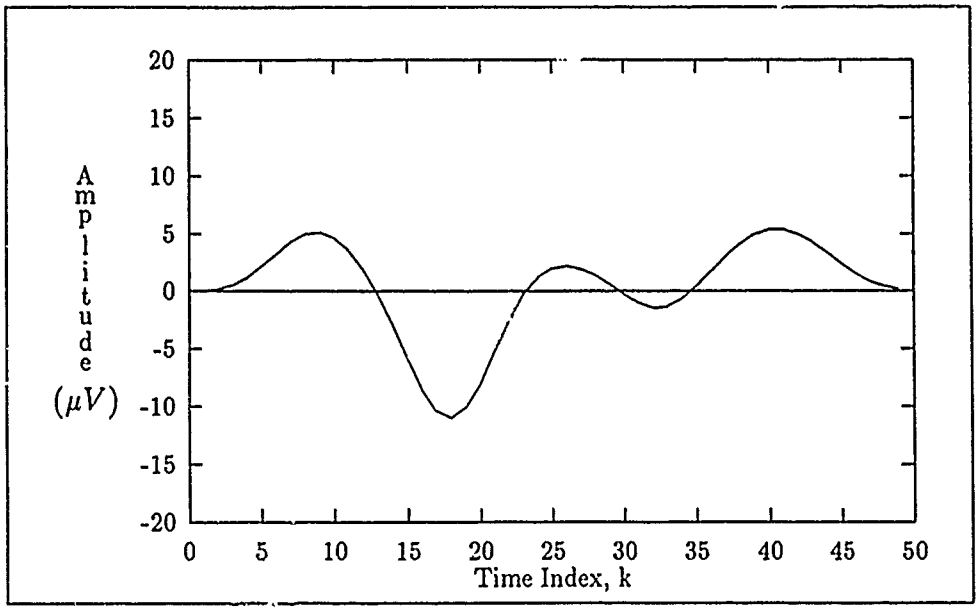

Figure 2.1. Ensemble Average of SDAT, M_j

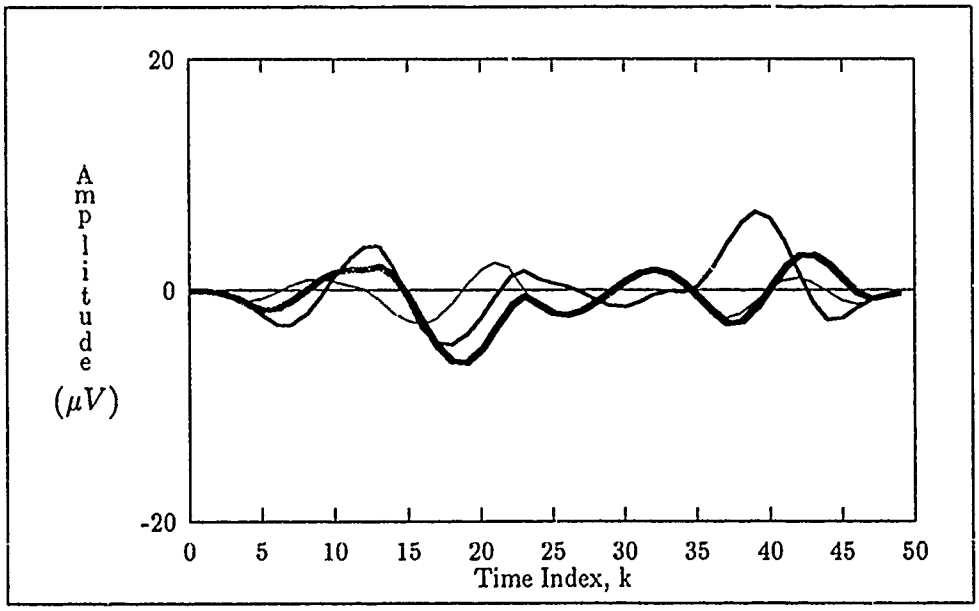

Figure 2.2. Example Q_j components of X_j

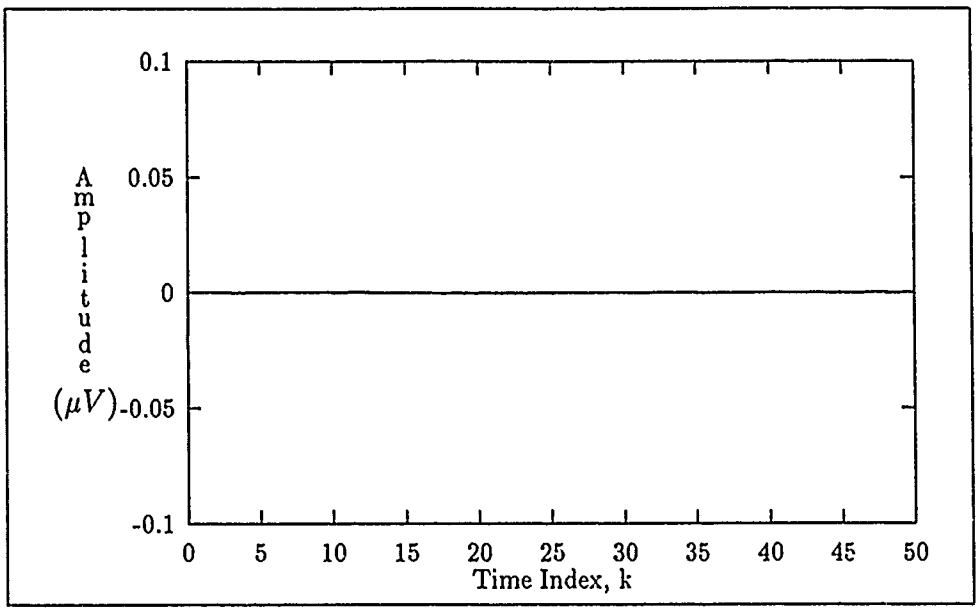

Figure 2.3. Time Sequenced Mean of Q_j

time varying. In addition, the plot shows that the initial response appears to repeat with a smaller amplitude at the end of the plot. In order to further characterize the jitter, the autocorrelation of the signal was calculated. This was done using Turbo-Basic, then the 50 point vector was imported to MathCad, a math software package. The plot of the autocorrelation function is shown in Figure 2.5. As expected, the function is even and has its maximum value at lag $\tau = 0$. In addition, one can observe significant correlation extending to approximately 20 data points.

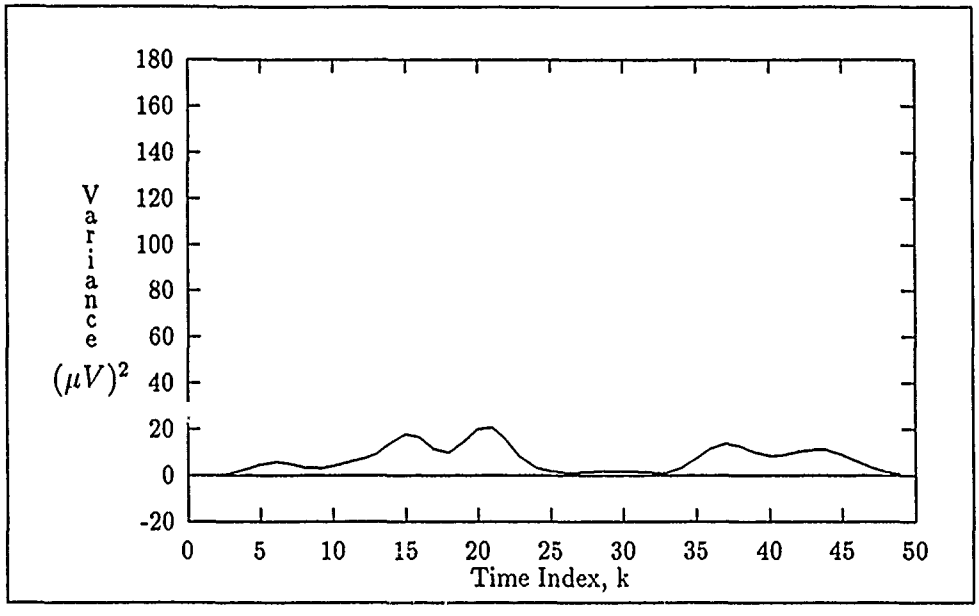

Figure 2.4. Variance of Q_j

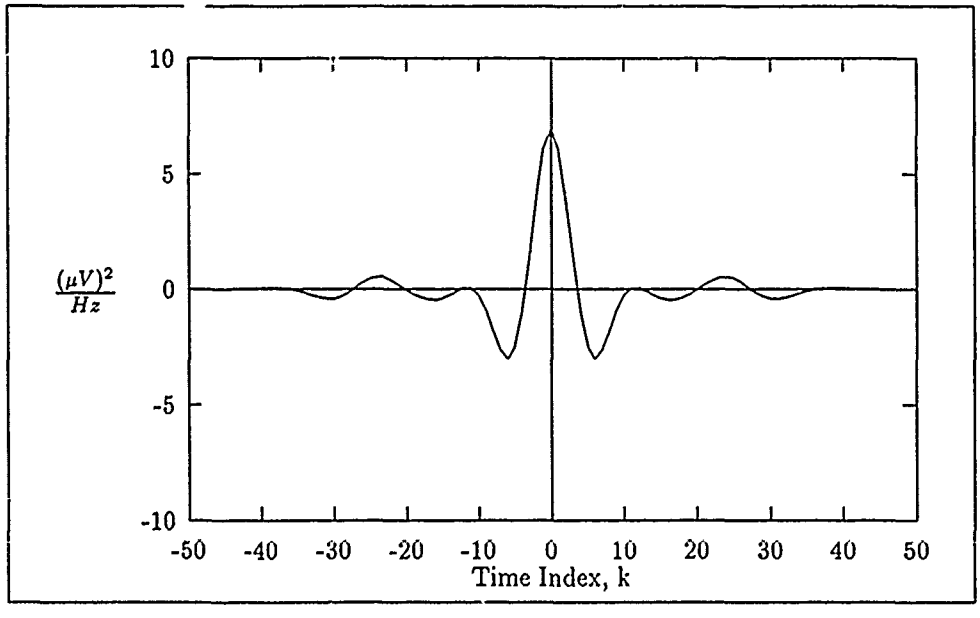

Figure 2.5. Autocorrelation of Q_j

2.2.3.3 Noise, N_j. The noise component of the signal was obtained simply by subtracting SDAT ($M_j + Q_j$) from DDAT ($M_j + Q_j + N_j$) which left 100 data vectors of 50 sample points each. The noise component of the signal is generally assumed to be zero mean with second order stationarity. To validate this assumption the ensemble average of N_j was calculated and is shown in Figure 2.6. Next, the ensemble mean-square or variance of N_j was calculated and plotted as a function of k (Figure 2.7). There is only a slight amount of variation in the signal variance estimate, thus validating the assumption that the signal's second order statistics are stationary. Thus, the noise component is wide sense stationary with zero mean.

2.2.3.4 Second Order Statistics of X_j. As stated earlier, X_j is the sum of three separated components: signal, jitter, and noise. A quantity of interest is the expected value of x_k^2 which is the following:

$$\begin{aligned} E[x_k^2] &= E[(m_k + q_k + n_k)^2] \\ &= E[m_k^2 + q_k^2 + n_k^2 + 2m_kq_k + 2q_kn_k + 2m_kn_k] \\ &= E[m_k^2] + E[q_k^2] + E[n_k^2] \\ &\quad + E[2m_kq_k] + E[2q_kn_k] + E[2m_kn_k] \end{aligned} \tag{2.3}$$

For the moment, assume the signal, jitter, and noise are uncorrelated which allows the cross product terms to be separated into the product of their individual expected values. Then, using the results from the previous sections that both Q_j and N_j have zero mean, the equation simplifies to the following:

$$E[x_k^2] = E[m_k^2] + E[q_k^2] + E[n_k^2] \tag{2.4}$$

The assumption that the signal, jitter, and noise are uncorrelated is now addressed. If the signal components are uncorrelated, then the sum of the individual

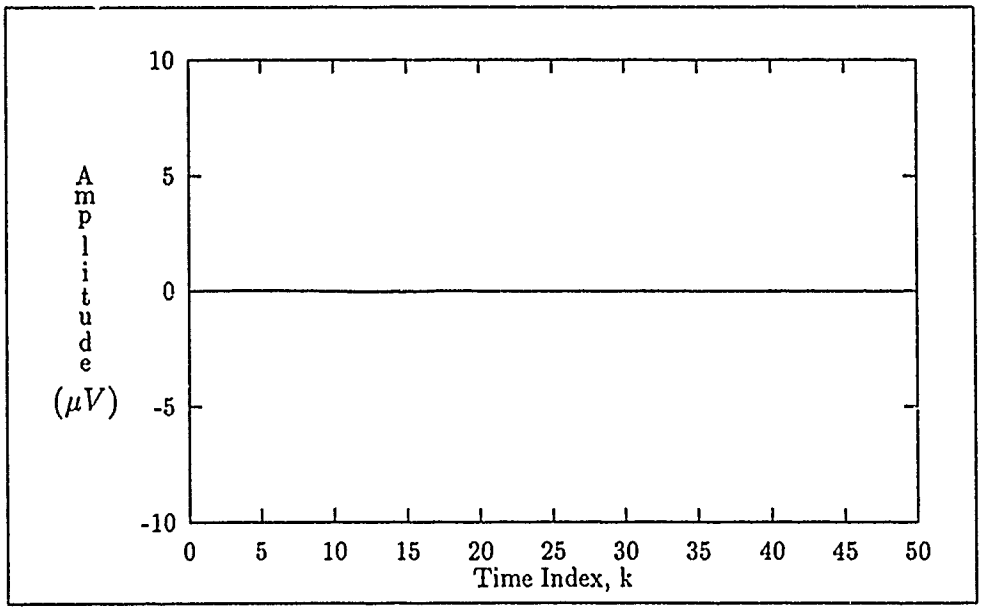

Figure 2.6. Ensemble Average of N_j

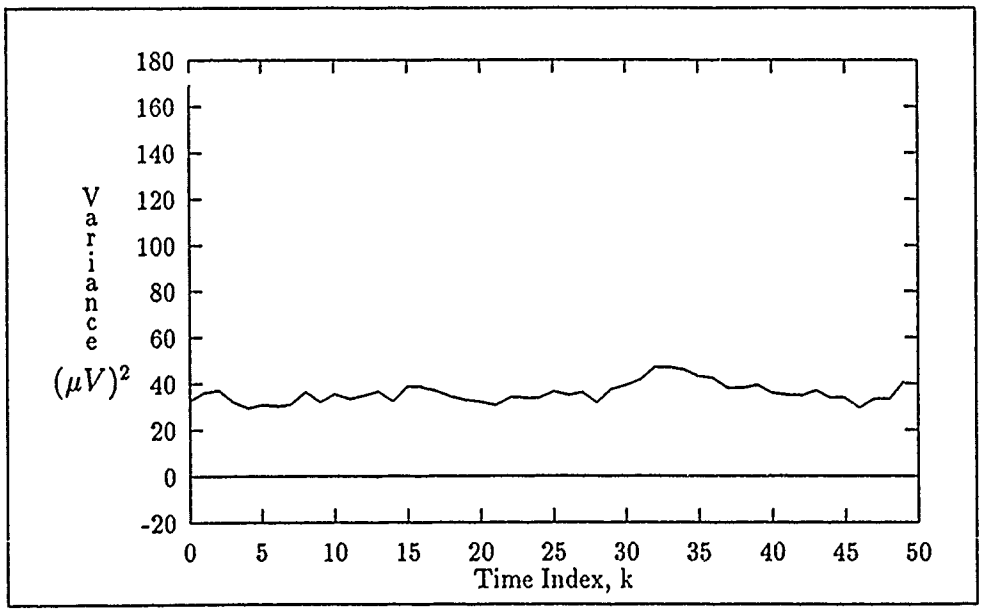

Figure 2.7. Variance of N_j

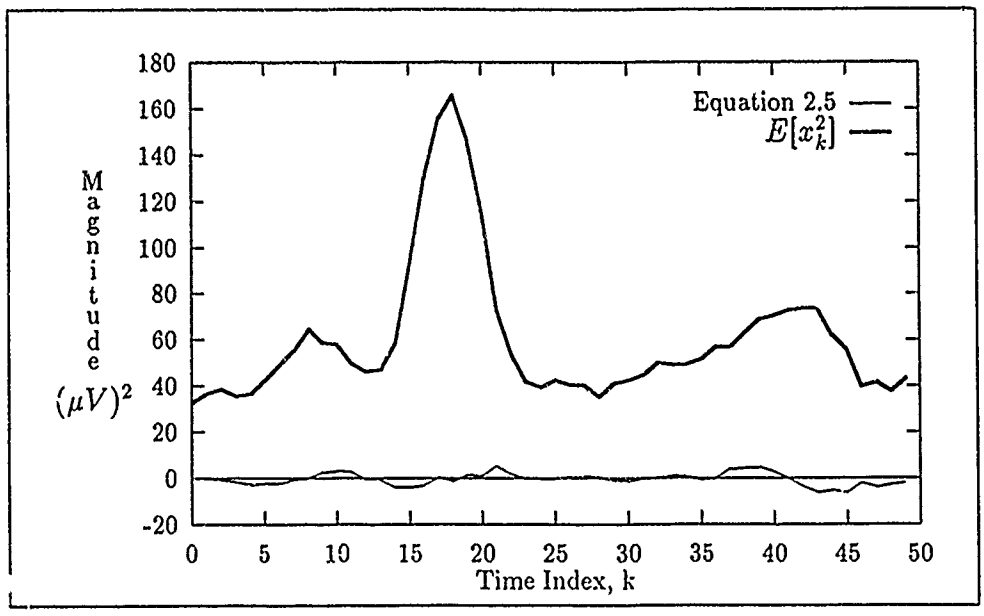

Figure 2.8. Comparison of $E[x_k^2]$ and $E[m_k^2] + E[q_k^2] + E[n_k^2]$

second order statistics will produce $E[x_k^2]$ which then implies the following:

$$E[x_k^2] - (E[m_k^2] + E[q_k^2] + E[n_k^2]) = 0 \tag{2.5}$$

To verify that indeed the signal, jitter, and noise components are uncorrelated, the difference in Equation 2.5 was formed from using the actual data vectors, then squared, and plotted as a function of k (see Figure 2.8). The plot indicates that there is a slight difference between the actual ensemble average of the signal and the summation of the individual components. However, the difference is minimal and the initial assumption that the individual components are uncorrelated is essentially valid.

2.2.3.5 Frequency Analysis of M_j and Q_j. In attempting to separate the jitter and the signal, one might try to use classical digital filtering techniques provided the frequency components did not overlap. However, a comparison of the FFT for M_j and Q_j reveals that there is significant frequency overlap between the

signals. Figure 2.10 is the FFT of autocorrelation of M_j and Figure 2.11 is the FFT of the autocorrelation of Q_j. Both data files were padded with zeros in order to meet the length requirements of 2^N where $N = 7$. Using a rectangular window, the FFT was performed on the individual data arrays using MathCad. It was also observed that M_j contains more power in relation to Q_j and there was frequency overlap of the two signals.

2.2.4 Summary. The EF is a magnetic field generated by the flow of ionic current along a nerve path. The source of the EF signal is modeled as a dipole which induces the flow of current along a thin wire. The signal is of very low power and is collected using a SQUID.

With the simulated EP signal modeled as shown in Equation 2.1, statistical analysis revealed many interesting characteristics of the simulated data signal. First, the average response signal, M_j, is nonstationary. Second, Q_j has a zero mean with nonstationary second order statistics and N_j is wide sense stationary with zero mean. Using the fact that Q_j and N_j are zero mean and the signal components are uncorrelated allowed Equation 2.3 to be simplified to

$$E[x_k^2] = E[m_k^2] + E[q_k^2] + E[n_k^2] \tag{2.6}$$

Finally, the frequency components of M_j and Q_j were shown to overlap which would not allow the use of filtering to separate the jitter and the response signal. The major point to bring out is that the EP signal is nonstationary and can be modeled as the sum of three uncorrelated components: the average response, jitter, and noise.

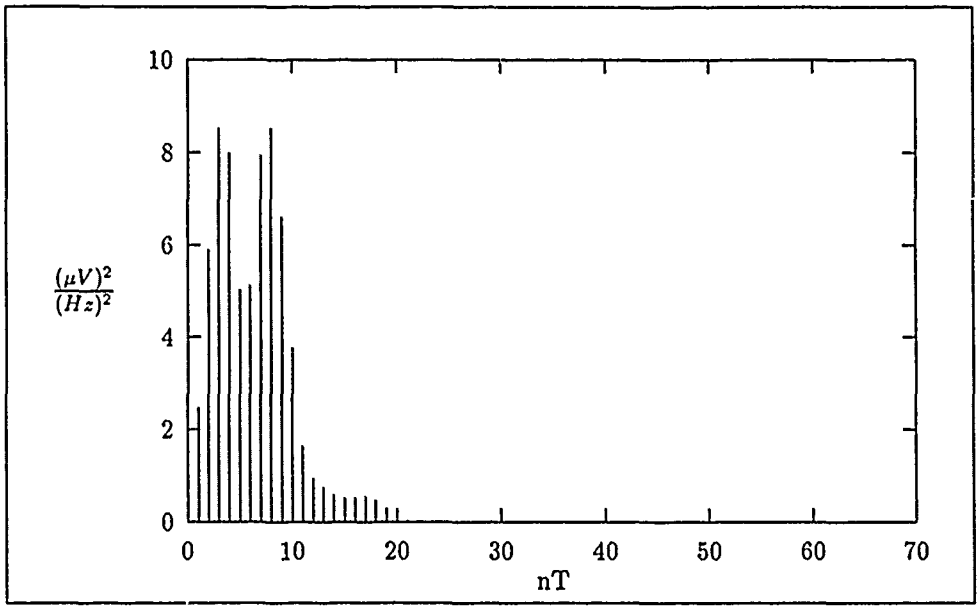

Figure 2.9. FFT of the Autocorrelation of M_j

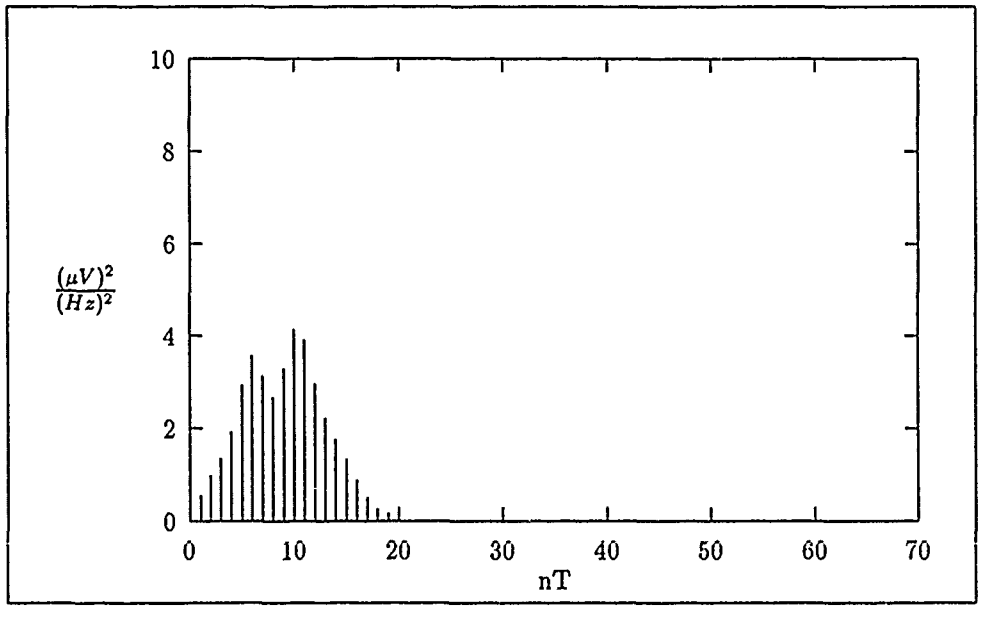

Figure 2.10. FFT of the Autocorrelation of Q_j

2.3 *The LMS Adaptive Filter*

This section presents a brief introduction to the theory of adaptive filters followed by a discussion of the Ferrara TSAF. The basic theory of the LMS adaptive filter is a building block used in the discussion on the Ferrara TSAF.

2.3.1 Definition. In general, an adaptive filter is a self-adjusting and time varying signal processor. The state of the filter is continually adjusted as a result of the input signal(s) and in some cases the output signal (10:5-9). Figure 2.11 illustrates a closed-loop adaptive filter which is the type commonly used in signal processing and is considered here. The input signal is x_k and the desired signal is d_k. The error signal, ϵ_k, is formed from the difference of the filter output, y_k, and the desired signal. The subscript k indicates the quantities are discrete values and k is the time index. The characteristic of a closed-loop filter is that the error signal is passed through some "Adaptive Algorithm" and then fed back to the processor (10:5). For this chapter, the filter theory is based on a causal filter and the commonly used least-mean-square (LMS) algorithm.

2.3.2 Overview of the LMS Filter. The processor of the adaptive filter is composed of a linear combiner and delay elements as shown in Figure 2.12 and is generally referred to as an adaptive linear combiner (10:15). The time index is k, the value of the zero weight at time k is w_{0k}, and the size of the filter is $L+1$ assuming a causal filter. The values of the weights are contained in the weight vector which is defined as follows:

$$W_k = [w_{0k}\, w_{1k} \ldots w_{Lk}]^T \tag{2.7}$$

The weight vector gives the state of the filter at time k. The filter output signal, y_k, will be the sum of all the delayed values of the input signal multiplied by their

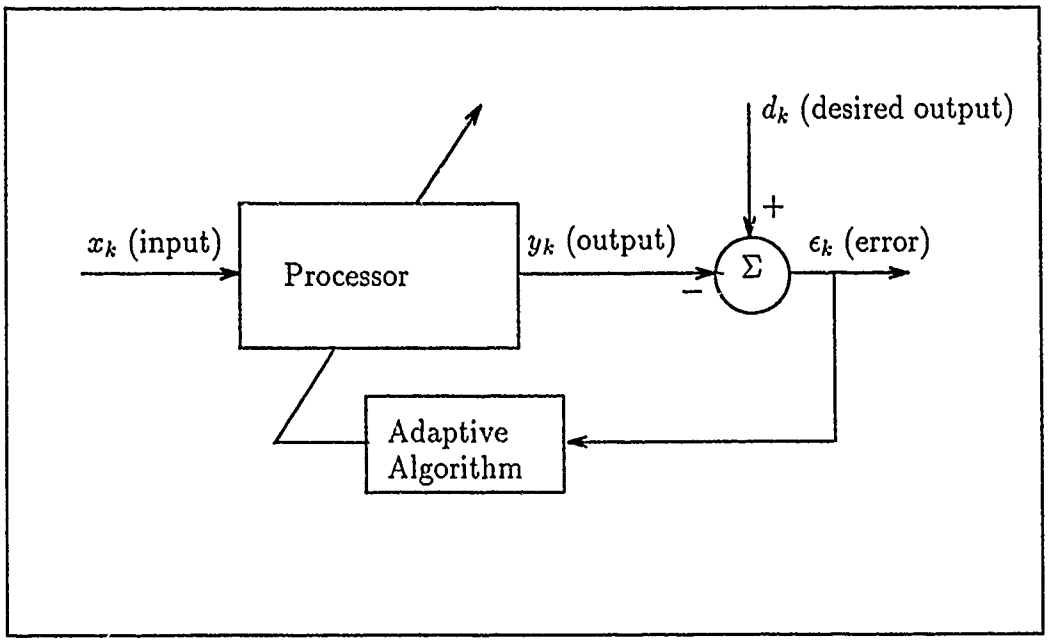

Figure 2.11. Adaptive Filter (3:9)

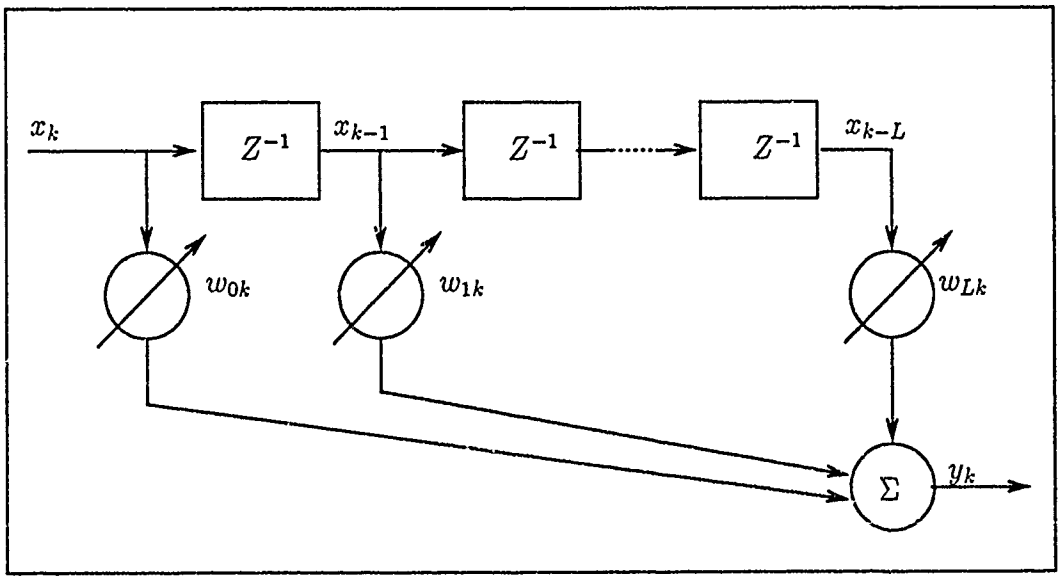

Figure 2.12. Adaptive Linear Combiner (3:17)

corresponding weight or

$$\begin{aligned} y_k &= \sum_{l=0}^{L} x_{k-l} w_{lk} \\ &= X_k^T W_k \end{aligned} \tag{2.8}$$

where $X_k = [x_k\, x_{k-1} \ldots x_{k-L}]^T$ (10:17).

The final signal to define is the error signal. As stated earlier, the error signal is the difference between the filter output and the desired signal or

$$\begin{aligned} \epsilon_k &= d_k - y_k \\ &= d_k - W_k^T X_k \end{aligned} \tag{2.9}$$

where the input signal is X_k, the desired signal is d_k, and the weight vector is W_k. The next step is to determine the value of W_k to optimize the performance of the filter.

The optimum performance for the LMS adaptive filter is realized when the mean squared error or $E[\epsilon_k^2]$ is minimized. Ideally, the optimum performance is achieved when $E[\epsilon_k^2] = 0$. However, this is seldom achievable in real applications and sometimes not desirable as we will see in the next section. The goal now is to determine how to express the performance of the filter mathematically and derive an expression for the optimum weight vector, W^*. Note that the k time subscript is dropped which assumes a stationary solution. The statistic of interest is the expected value of the squared error (the mean square error) which is defined as follows:

$$\begin{aligned} E[\epsilon_k^2] &= E[d_k + W_k^T X_k X_k^T W_k - 2 d_k X_k^T W_k] \\ &= E[d_k] + E[W_k^T] E[X_k X_k^T] E[W_k] - 2E[d_k X_k^T] E[W_k] \\ &= E[d_k] + W_k^T E[X_k X_k^T] W_k - 2E[d_k X_k^T] W_k \end{aligned} \tag{2.10}$$

Two significant assumptions were made to derive Equation 2.10. First, the weight vector and the input signal vector are uncorrelated. This allows the initial equation to be written as shown on the second line. Second, the filter has reached some point in time where $E[W_{k+1} - W_k]$ equals the zero vector. In other words, the weight vector has converged to a solution and is treated as a constant with the k subscript dropped. The final form of Equation 2.10 is

$$E[\epsilon_k^2] = E[d_k] + W^T R W - 2P^T W \tag{2.11}$$

where R is called the input autocorrelation toeplitz matrix and is defined as

$$R = E[X_k X_k^T] = \begin{bmatrix} \phi_{xx}(0) & \phi_{xx}(-1) & .... & \phi_{xx}(-L) \\ \phi_{xx}(1) & \phi_{xx}(0) & ... & \phi_{xx}(1-L) \\ \phi_{xx}(2) & \phi_{xx}(-1) & .... & \phi_{xx}(2-L) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{xx}(L) & \phi_{xx}(L-1) & ... & \phi_{xx}(0) \end{bmatrix} \tag{2.12}$$

and $\phi_{xx}(n) = E[x_k x_{k+n}]$. P is the cross correlation vector and is defined as

$$P = E[d_k X_k] = \begin{bmatrix} \phi_{dx}(0) \\ \phi_{dx}(-1) \\ \vdots \\ \phi_{dx}(-L) \end{bmatrix} \tag{2.13}$$

where $\phi_{dx}(n) = E[d_k x_{k+n}]$. Assuming that the input signal and desired signal are stationary, R and P are constant and, therefore, they need no time subscript. However, as it will be shown later, if the input signals are not stationary, the autocorrelation matrix and the cross correlation vector will change with time. Equation 2.11 is the performance equation of the LMS filter, also referred to as the mean-square error (MSE) equation, and describes a performance surface which the filter searches to

find the global minimum (10:20). The existence of a global minimum is guaranteed by the characteristics of the quadratic equation. The final step is to take the gradient of $E[\epsilon_k^2]$, set it equal to zero, and solve for the optimum weight vector. First, the gradient vector, ∇, is defined as follows:

$$\nabla_w F = [\partial F/\partial w_0 \; \partial F/\partial w_1 .... \partial F/\partial w_L]^T \tag{2.14}$$

where F is a function in w and ∇_w is the gradient operator and performs the first partial derivatives on the argument with respect to w_k (6:142). Next, applying the ∇_w operator to Equation 2.11 and solving for the minimum yields the following:

$$\begin{aligned} 0 &= \nabla_w E[d_k] + \nabla_w W^T R W - \nabla_k 2P^T W \\ &= 0 + \nabla_w (W^T)(RW) - \nabla_w 2P^T W \\ &= 2RW - 2P \end{aligned} \tag{2.15}$$

Remembering that the ∇_w can be written as a column vector (Equation 2.14), the result was a straight forward manipulation working the matrix equations from left to right and then applying the ∇_w operator. Solving Equation 2.15 for the optimum weight vector, W^* yields:

$$W^* = R^{-1}P \tag{2.16}$$

It is important to note that this result assumes R is invertible (10:22). Finally, substituting Equation 2.16 into Equation 2.11 for W produces the equation for the minimum mean-square error which is the following:

$$\xi_{min} = E[d_k^2] - P^T R^{-1} P \tag{2.17}$$

Equations 2.16 and 2.17 show that the optimum weight vector depends on the inverse of the input autocorrelation matrix and the cross correlation vector. Given a wide sense stationary input signal, the R matrix can be calculated and will have

constant values. The P vector is calculated by performing the cross correlation of X_k and d_k. As the correlation between the input signal and the desired signal increases, the MSE decreases. In other words, the adaptive filter searches the performance surface for the global minimum and in doing so maximizes the correlation between X_k and d_k (12). Figure 2.13 is an example of a quadratic performance surface. The vertical axis is the MSE and the horizontal axis are the weight values. There are only two weights in this example. The bottom of the bowl represents the optimum solution as defined by Equation 2.16. If the slope of the surface was shallow, the resulting gradient would be small and the weight vector would converge slowly towards the optimum solution represented at the bottom of the bowl. However, if the slope of the sides is steeper, the resulting gradient be larger and will drive the weights towards the optimum solution faster. The main point is that the shape of the surface determines the magnitude of the resulting gradient (12).

Another curve of interest for the LMS is the learning curve which is shown in Figure 2.14. This shows the effect to the MSE as the filter weights change to adapt to the input signals. Also shown is ξ_{min} which is the horizontal line on the plot and represents the error once the filter has reached the optimum solution as defined in Equation 2.16 (10:51). In practice, the filter will seldom reach the ideal error because of adaptation noise which biases the filter solution.

The critical assumption used throughout this development is that the input signal is at least wide sense stationary. However, the EP signal is nonstationary. Given the input signal is nonstationary, the performance surface and thus the minimum point is now changing with time which then implies that the optimum weight vector is also changing in time (3:519). In addition, the autocorrelation matrix and the cross correlation vector are no longer fixed and the assumptions used in Equation 2.10 no longer apply. The next section discusses a nonstationary input to the LMS filter in terms of the weight vector update equation.

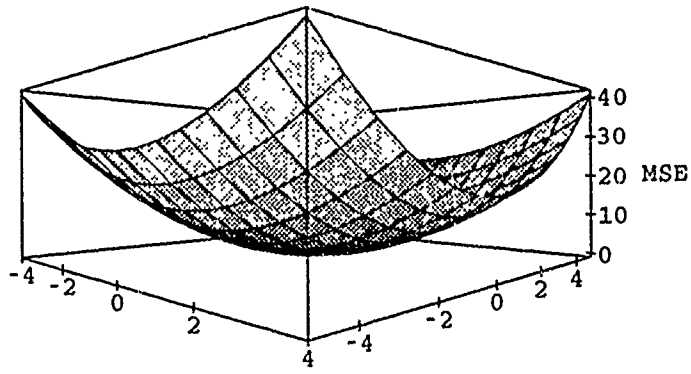

Figure 2.13. Performance Surface

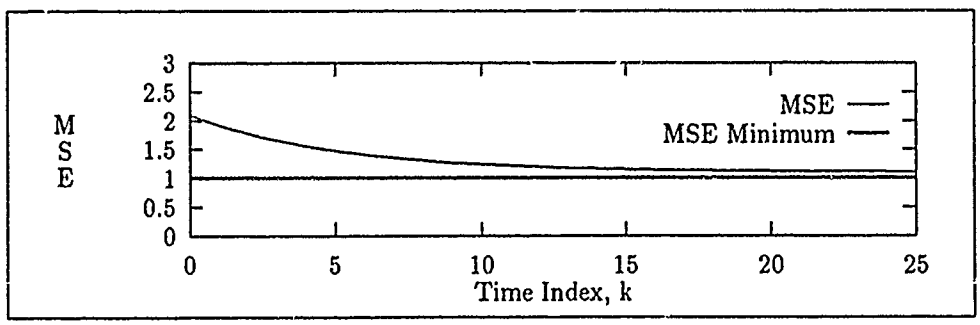

Figure 2.14. Learning Curve for LMS Filter

2.3.3 Nonstationary Input. The filter vector for the LMS algorithm is modified using the following equation:

$$W_{k+1} = W_k + 2\mu\epsilon_k X_k \tag{2.18}$$

where μ is the gain constant and is used to adjust the speed at which the filter adapts as well as the stability of the adaptation (10:100). Equation 2.18 shows that the next weight vector, W_{k+1}, depends on the present weight vector plus a scaled estimate of the error. Therefore, μ determines the proportion of the error which is included in the update (3:520). For input signals that are slowly varying in time, the μ can be increased to allow the filter to track the non-stationarity. However, as μ is increased the next weight vector is composed of a higher estimate of the error. In other words, increasing μ will cause the weight vector to vary around the actual solution and in effect, add adaptation noise to the weight vector (13:34). Widrow points out that noise generated due to the "lag" of the filter response is additive to the noise generated by μ. This "lag" noise is due to the filter trying to track a performance surface which is changing in time (11:1151).

The point of all this is to show that an LMS adaptive filter is not designed to process signals with rapidly varying statistics. However, the Ferrara time sequenced adaptive filter uses the LMS adaptive concept to process a certain class of nonstationary signals.

2.3.4 Summary. The LMS adaptive filter searches a performance surface to find the global minimum and thus maximizes the correlation between the input signal and the desired signal. The optimum weight vector for a stationary signal is given by Equation 2.16 and assumes the input signals are at least wide sense stationary. The adaptive filter performance is degraded if the input signals statistics change rapidly with time and may not converge to a fixed optimum weight vector. Therefore, use of the Ferrara TSAF is considered and is the topic of the next section.

2.4 *Ferrara Time Sequenced Adaptive Filter*

This section defines the TSAF and overviews the theory based on the LMS theory presented in the previous section. A good deal of time will be spent developing the optimum weight solution for the TSAF as part of this thesis is a direct application of the TSAF.

2.4.1 Definition. The Ferrara TSAF is an adaptive filter designed to process nonstationary signals which have statistics which repeat in time or are cyclostationary (2:21-22). The advantage of a TSAF is that it "...allows the weight vector to change freely in time in order to accommodate rapid changes in the statistics of a certain class of nonstationary signals, while allowing slow precise adaptation" (2:22). The input signal for the TSAF is required to have a finite set of statistics that repeat in time (2:22). The importance of this requirement will become evident in the following discussions.

2.4.2 Overview of the TSAF. The TSAF can be viewed as a bank of time multiplexed LMS adaptive filters in parallel with only one filter active at a time (Figure 2.15). Given that the input signal's statistics repeat, the signal is assumed to have a finite set of performance surfaces between repetitions. In addition, the performance surfaces repeat following the same sequence during each repetition interval. The TSAF is set up such that the filter sequences through the individual performance surfaces at the start of each repetition (2:22-24). For the EP and EF signals, the start of the repetition is the index time associated with the visual stimuli. Each EF sample has 50 data points and therefore the filter will initially have 50 separate performance surfaces and 50 separate optimum weight vectors.

We are now ready to modify the equations presented for the LMS adaptive filter to account for the TSAF. Equation 2.16 was valid for a wide sense stationary signal and produced a single optimum weight vector for the single performance surface. Now there are multiple performance surfaces and multiple input autocorrelation

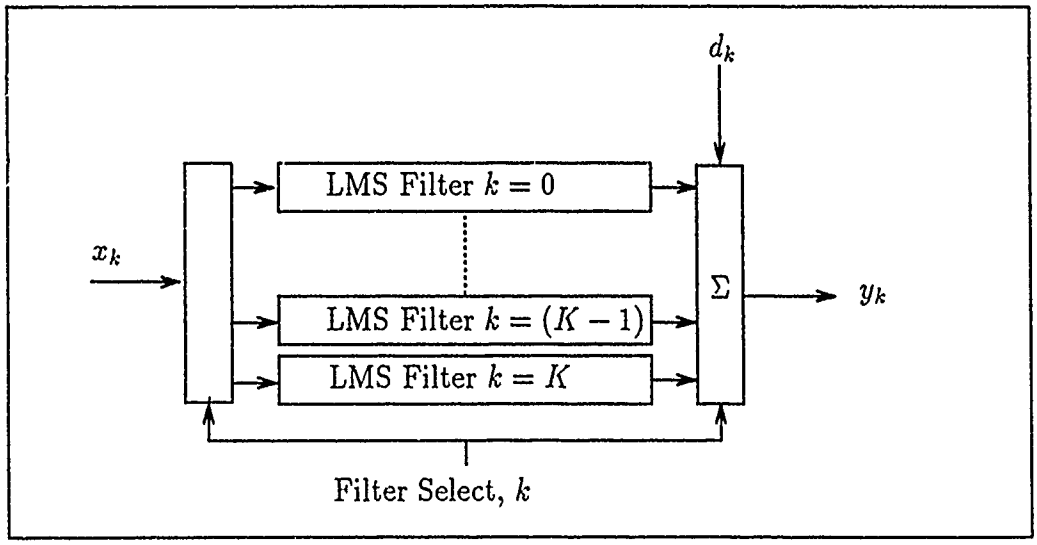

Figure 2.15. Time Sequenced Adaptive Filter

matrices and cross correlation vectors. The result is that there will be multiple optimum weight vectors. Thus, the optimum weight vector for the TSAF is defined as follows:

$$W_k^* = R_k^{-1} P_k \tag{2.19}$$

The k index shows the effect of the time sequence on Equation 2.16 (2:27). This result is not surprising if the TSAF is viewed as a bank of LMS adaptive filters in parallel with only one filter active for each point in time or for each value of k. For this thesis, each of the 50 data points will have a corresponding weight vector which is updated as the filter moves "across" the data ensemble. It is now obvious that if the set of performance surfaces was not finite (i.e. the statistics do not repeat), the filter would have an infinite number of optimum weight vectors and would not be realizable.

2.4.3 Optimum Augmented Solution. The development of the optimum weight vector solution up to this point has assumed that the input signal does not have a bias. This may not be true in general and for the EP signal is indeed not the case.

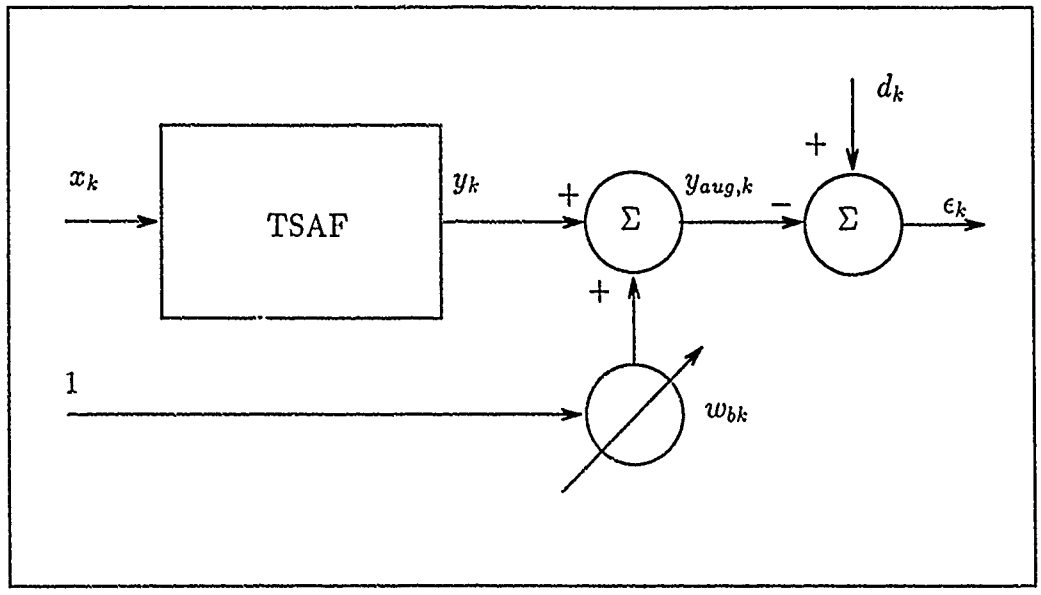

Figure 2.16. TSAF Filter With Bias Weight

It is now appropriate to present the time sequenced optimum weight vector solution with a time-sequenced bias weight present to compensate for the time-sequenced non-zero mean of the input signal. The notation will follow that used by Williams in his development of the optimum weight vector (13).

Figure 2.16 shows the filter with the time-sequenced bias weight added. The error signal is still formed from the difference of the desired signal and a new output signal, $y_{aug,k}$, which includes the bias weight and the filter output y_k. The augmented output signal is defined as

$$\begin{aligned} y_{aug,k} &= w_{b,k} + y_k \\ &= w_{b,k} + W_k^T X_k \\ &= \tilde{W}_{aug,k}^T \tilde{X}_{aug,k} \end{aligned} \tag{2.20}$$

where

$$X_{aug,k} = [1 \; X_k]^T \tag{2.21}$$

$$W_{aug,k} = [w_{b,k} \; W_k]^T \tag{2.22}$$

The augmented vectors in Equations 2.21 and 2.22 include the unity input to the bias weight and the value of the bias weight at time k (13:39).

Now that the augmented input and weight vectors have been derived, the next step is to determine the form of the optimum weight vector solution. Equation 2.19 is still the form of the solution. However, a new autocorrelation matrix and cross correlation vector must be defined. The augmented autocorrelation matrix $R_{aug,k}$ is defined as

$$R_{aug,k} = E[X_{aug,k} \, X_{aug,k}^T] \tag{2.23}$$

Then using Equation 2.21 and substituting directly into Equation 2.23, the matrix is now written as

$$R_{aug,k} = \begin{bmatrix} E[1] & E[X_k^T] \\ E[X_k] & E[X_k X_k^T] \end{bmatrix} \tag{2.24}$$

Recall that the input signal can be modelled as the sum of three independent components: the average signal response, the jitter, and the pre-stimulus noise. Then $E[X_k] = M_k$ where M_k is the average signal response as defined in Section 2.2.3. Using this and Equation 2.12 the final form of augmented autocorrelation matrix is

$$R_{aug,k} = \begin{bmatrix} 1 & M_k^T \\ M_k & R_k \end{bmatrix} \tag{2.25}$$

The final hurdle is to define the new cross correlation matrix. This is fairly straight forward as shown in the following:

$$P_{aug,k} = E[d_k X_{aug,k}]^T$$

$$
\begin{aligned}
&= E[d_k[1 \;\; X_k]]^T \\
&= [m_{d,k} \;\; E[d_k X_k]]^T \\
&= [m_{d,k} \;\; P_k]^T
\end{aligned} \tag{2.26}
$$

Equation 2.21 was directly substituted into the second line with $E[d_k] = m_{d,k}$ used in the third line. Putting it all together, the optimum weight vector solution,which includes the bias weight, is

$$
W^*_{aug,k} = \begin{bmatrix} 1 & M_k^T \\ M_k & R_k \end{bmatrix}^{-1} \begin{bmatrix} m_{d,k} \\ P_k \end{bmatrix} \tag{2.27}
$$

The time index k is present to remind us that the optimum solution is time varying. Both Williams and Ferrara further reduce Equation 2.27 to gain additional insight into the effect of the bias weight on the optimum solution and how it removes the input signal mean (2) (13). The derivation is provided in Appendix B with only the results presented here. By performing the vector multiplication in Equation 2.27 and performing some algebraic manipulation, the optimum weight vector solution can be written as follows:

$$
W^*_{aug,k} = \begin{bmatrix} w_{b,k} \\ W_k \end{bmatrix} = \begin{bmatrix} m_{d,k} - M_k^T F_k^{-1} Q_{d,k} \\ F_k^{-1} Q_{d,k} \end{bmatrix} \tag{2.28}
$$

where $F_k = E[(Q_k + N_k)(Q_k + N_k)^T]$ and is the autocorrelation matrix of the zero mean signal. $Q_{d,k} = E[q_{d,k} Q_k]$ and is the expected value of the correlated signal components of the input signal and the desired signal. In looking at Equation 2.28 the bias weight has removed the mean component of the input signal scaled by $F_k^{-1} Q_{d,k}$. The output must be corrected to reflect the incorrect alteration made to the input signal. This correction is taken care of by the bias weight adding on an estimate of the correct mean, $m_{d,k}$ (2:49). Another way to see this is to look at the output of the filter in terms of the optimum weight solution.

The filter output signal, $y_{aug,k}$, can be written in terms of the optimum filter solution or

$$\begin{aligned} y_{aug,k} &= w_{b,k} + y_k \\ &= w_{b,k} + X_k^T W_k^* \\ &= m_{d,k} - M_k F_k^{-1} Q_{d,k} + X^T F_k^{-1} Q_{d,k} \\ &= m_{d,k} + (X_k^T - M_k^T) F_k^{-1} Q_{d,k} \end{aligned} \tag{2.29}$$

The filter output equation "...suggest that the filter taps generate an estimate of both the mean and the correlated stochastic components of the desired signal while the bias weight removes the mean estimate generated by the filter taps and replaces it with the correct mean estimate"(13:52).

Ferrara further points out that the input signal mean also shows up in the minimum mean-square error and will increase the MSE of the filter if not removed. This is especially true if the power of the mean component is high relative to the other signal components (2:50). Therefor, if an input signal has a non-zero mean, the bias must be included to reduce the error term generated by the filter. In other words, the signal bias will offset the performance surface of the filter and the bias weight will shift the surface to the desired position.

2.4.4 Summary. The Ferrara TSAF can be viewed as a bank of LMS filters time multiplexed and in parallel with only one filter active at a time. The other key point made in this section is that if an input signal has a bias, the bias will shift the solution away from the desired optimum. In addition, the filter configuration proposed by Ferrara with the bias weight added to the output of the filter was analyzed with a presentation of the optimum weight vector solution in terms of the bias weight and the Williams signal model.

2.5 *Chapter Summary*

Several major points presented in this chapter are summarized before proceeding to the next chapter. The EF and EP signals are a class of nonstationary signals produced by the brain in response to auditory and/or visual stimuli. The EF signal is modeled as a magnetic field generated by the flow of current along a thin wire and is detected using a special device called a SQUID. The EP signal is the change in potential due to the visual stimuli. This signal has been studied in great detail and, for the purposes of this thesis, has been modeled as the sum of three uncorrelated components: average signal response, signal jitter, and noise. This thesis considers the application adaptive filter theory to the processing of these nonstationary signals which are buried in strong background noise.

The brief discussion presented on adaptive filters revealed that while the LMS filter can adapt to signals with slowly changing statistics, the classical LMS filter is not designed to handle signals with rapidly changing statistics as increasing the speed of the adaptation adds noise to the solution. This warrants exploring the use the Ferrara TSAF which is designed to process cylostationary signals (i.e signals that have statistics which repeat in time). Furthermore, the EP signal has a non-zero mean and some method must be used to compensate for the bias to reduce the MSE of the filter and thus increase the accuracy of the optimum weight solution. Ferrara suggests one method for removing the mean using a bias weight at the output of the TSAF filter which is also time varying. The next chapter discusses the TSAF and mPa algorithms along with a modified TSAF filter which removes the signal bias before the TSAF filter.

III. Filter Implementation and Verification

3.1 Introduction

The previous chapter presented the reasons for selecting the TSAF and the theory behind the TSAF. The next logical steps are to implement the filter in software and then test the program. Even after the filter implementation is verified, the concept of estimating human EF using human EP must somehow be validated. Therefore, there are five major goals for this chapter which are discussed next.

1. Define the equations and algorithms implemented in software. This section is intended to familiarize the reader with the algorithms as they are implemented in software and present an overview of how the program is organized.
2. Test individual sub-components to ensure they are error free. The purpose of testing individual components is to quickly isolate any problems that might exist with the code. The algorithms tested in this section are common to both the LMS and the mPa and are the following:
   - μ_k Update Algorithm.
   - Bias Weight Update Algorithm.
   - Filter Weight Update Algorithm.
3. Test the LMS filter to ensure it is error free. At this point the filter is fully assembled with all the sub-components and tested with the LMS algorithm. There are two test performed in this section. The first test uses noiseless input signals which allows for a theoretical analysis. The second test stresses the filter by adding noise to the input signal.
4. Test the mPa filter to ensure it is error free. The mPa has a unique input, the P_n vector, which is an estimate of the cross correlation statistics of the noise

components. Therefore, this section first tests the P_n estimator and then the mPa filter which uses the output of the estimator.

5. Validate the concept of estimating human EF using human EP. This section uses simulated EP and simulated EF signals to test the filter in the configurations used in Chapter 4 for estimating the human EF. Simulated noise is created for this section based on the cross correlation statistics of the human EEG and human MEG and the autocorrelation of the human MEG. The use of these simulated noise files and simulated signal files provides an elementary model of the processing performed in Chapter 4. Therefore, this section tests the TSAF and the concept of using the filter with human data.

Before proceeding to the filter overview, a brief section defining the notation used in this chapter is presented next.

3.2 Notation

Several subscripts were used in this chapter and they are summarized below:

- j identifies a specific data vector contained in the data ensemble. For example, X_j is the j^{th} data vector. j ranges from $0 \ldots (M-1)$ where M is the number of data vectors in the ensemble.
- k identifies a specific sample at time k within a given vector. k ranges from $0 \ldots (N-1)$ where N is the number of data points in the vector.
- aug indicates the original vector is augmented with a bias weight. See the TSAF filter solution presented in Chapter 2.
- b indicates the variable is a bias weight.
- d indicates the variable is associated with the desired input signal.
- x indicates the variable is associated with the input signal.
- p indicates the variable is associated with the mPa and P_n vector.

The TSAF uses either the LMS algorithm or the mPa as will be discussed. To simply the notation, $TSAF_{LMS}$ indicates the LMS algorithm is selected and $TSAF_{mPa}$ indicates the mPa is selected.

3.3 $TSAF_{LMS}$ Filter

The $TSAF_{LMS}$ program performs three major functions. First, the program tracks and updates all the weights for each filter based on the LMS algorithm. Second, the degradation of the MSE due to the presence of a bias in the input signal is reduced using a bias weight. To speed the convergence of the filter, a modified TSAF filter is used with an additional bias weight preceding the TSAF. Third, the program allows the gain constant, μ, to adapt to different input signal energy levels as will be shown.

3.3.1 $TSAF_{LMS}$ Filter Update Algorithm. The LMS algorithm is the building block for the time sequenced algorithm used in this thesis and is presented here once again

$$W_{k+1} = W_k + 2\mu\epsilon_k X_k \tag{3.1}$$

k is the time index, μ is the gain constant, and ϵ_k is the error. The LMS algorithm is implemented using a linear combiner and is important because it is fairly straight forward to implement and does not require any off line processing (10:99-100). The important point to make is that the weight update for the LMS is done over time and thus a function of k. On the other hand, the TSAF filters are updated with the next data vector at the same point in time as will be shown next.

To implement the $TSAF_{LMS}$ algorithm, one simply extends the LMS algorithm such that there are as many weight vectors as discrete data points in a single data vector. In other words, given a data ensemble of 100 vectors each with 50 discrete sample points, the range on k is 0 to 49 and the $TSAF_{LMS}$ will have a separate filter for each value of k. Then each filter is updated with the next data vector at

the same point in time. This means that the $TSAF_{LMS}$ filters are updat' "across" the ensemble. This differs from the LMS in that the LMS algorithm is updated in time or "along" the ensemble which accounts for the additional subscript "j" in the $TSAF_{LMS}$ algorithm as shown in the following:

$$W_{j+1,k} = W_{j,k} + 2\mu_k \epsilon_k X_{j,k} \tag{3.2}$$

k is the time and filter index and j is the data vector index. Again, note that the weight vector is updated with the next data vector at the same point in time. The next step is to add in the bias weight and rewrite Equation 3.2 using the augmented vectors presented in the previous section. The $TSAF_{LMS}$ algorithm with the bias weight is then

$$W_{aug,j+1,k} = W_{aug,j,k} + 2\mu_k \epsilon_k X_{aug,j,k} \tag{3.3}$$

Expanding this equation shows the update algorithm for the bias weight as well as the $TSAF_{LMS}$ filter:

$$W_{aug,j+1,k} = \begin{bmatrix} w_{b,k} \\ W_k \end{bmatrix}_{j+1} = \begin{bmatrix} w_{b,k} \\ W_k \end{bmatrix}_j + 2\mu_k \epsilon_k \begin{bmatrix} 1 \\ X_k \end{bmatrix}_j \tag{3.4}$$

With the discussion moving towards the bias weight, it is appropriate to present the change to the $TSAF_{LMS}$ filter configuration.

Recall that the purpose of the bias weight is to reduce the MSE degradation caused by the presence of any bias in the input signal. Figure 3.1 shows a modified $TSAF_{LMS}$ filter designed to remove the bias of the input signal before the filter. The first thing to notice is that an additional time- sequenced bias weight, $w_{b,k}$, has been added to the front of the $TSAF_{LMS}$ filter. By using a two stage filter, the first stage removes the bias, m_k, and the second stage then operates on the jitter and noise components of the input signal or $q_k + n_k$ (13:114-115). Williams points out that removing the bias before the filter allows the use of a larger μ for faster convergence

as the filter does not see the additional energy contained in the mean component of the signal (13:81). This will be presented in more detail shortly.

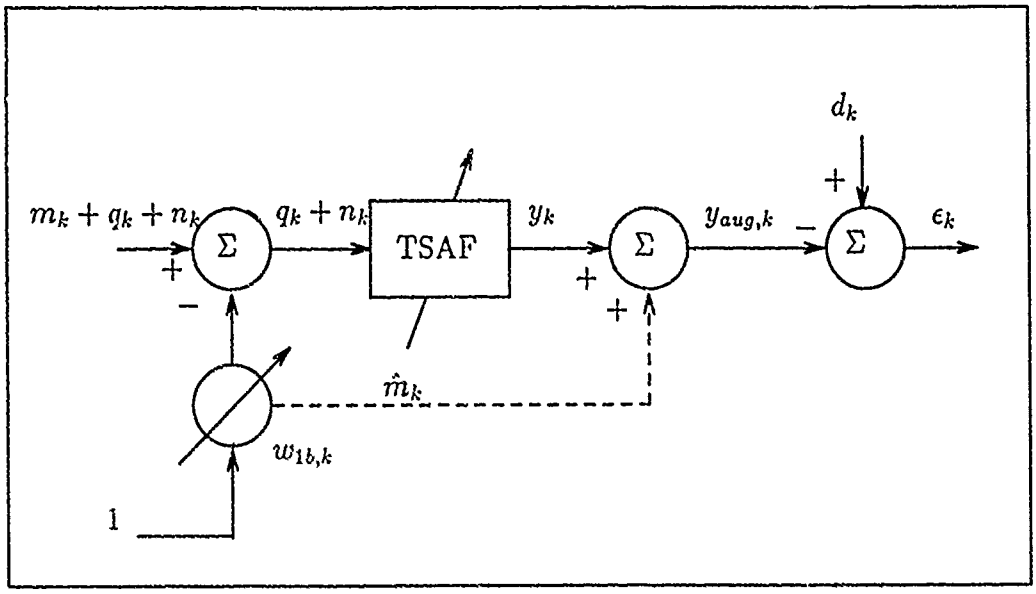

Figure 3.1. Two Stage TSAF

To remove the bias before the filter requires some knowledge of what the bias is. This knowledge will be learned over time by the bias weight using the following algorithm:

$$w_{b,j+1,k} = w_{b,j,k} + \frac{(2)(0.5)}{j+1}(x_{j,k} - w_{b,j,k}) \tag{3.5}$$

$j = 0 \ldots (M-1)$ and $k = 0 \ldots (N-1)$ where M is the number of data vectors and N is the number of points in each vector. Equation 3.5 is simply a recursive mean estimator and will converge to the ensemble average value for time k. An important note is that the value of $w_{b,k}$ is added back to the filter output. In addition, there is a bias weight for each time sequenced filter and when $w_{b,j+1,k} = w_{b,j,k}$, the bias weight vector will equal the ensemble average given a perfect adaptation. The last algorithm is the calculation of μ.

As stated earlier, the gain constant μ is determined adaptively for this thesis and in general is defined as

$$\mu = \frac{Misadjustment}{(L+1)(signal\ energy)} = \frac{M}{(L+1)(E[x_k^2])} \tag{3.6}$$

where $L+1$ is the number of filter taps (13:53). The misadjustment is "...a measure of how closely the adaptive process tracks the true Wiener solution ..." (10:110). As was presented earlier, for larger values of M and thus larger values of μ the adaptation will have more noise in the solution with faster adaptation. For this thesis, the misadjustment is fixed at 0.05 with the μ allowed to vary as a function of the filter size and signal energy. The reason for not fixing μ is to allow for optimum adaptation at various signal energy levels. Looking at Equation 3.2 one can see that if the energy of the $X_{j,k}$ vector is large, it will change the resulting update vector more than a smaller energy input vector for a fixed μ. However, if μ is updated with each new input vector based on the signal energy, a stable convergence will result for all the filters even though the each filter operates on different portions of the input signal. The μ is then updated according to the following recursive algorithm:

$$\mu_{j+1,k} = 0.95\mu_{j,k} + 0.05\frac{M}{(L+1)(E[x_k^2])} \tag{3.7}$$

The k subscript indicates there is a separate μ for each filter. The 0.95 essentially adds finite memory to the update and can be decreased if a shorter memory of past values is desired (13:53-54). The next step is to present how the $TSAF_{LMS}$ filter is implemented in software.

3.3.2 TSAF Software Overview. Each of the algorithms presented thus far are implemented in separate procedures which are then called from the driver or main program loop. The flow of the program for the $TSAF_{LMS}$ is presented below:

- Initialize variables, arrays, and vectors (only once).

- Load input signal and desired signal.
- Update the bias weight.
- Remove bias from the input signal.
- Calculate and update μ_k.
- Calculate output signal, y_k.
- Add bias to y_k forming $y_{aug,k}$.
- Calculate error, e_k
- Update weight vector.
- Loop if more data.

Appendix F contains the program listing with an introduction to the data size used and general program information. In general, the filter program ran several passes through the data ensemble to let the bias weight learn the ensemble average. The bias weights were then frozen and the data ensemble passed through again to let the filter weights adapt using zero mean filter inputs.

3.4 $TSAF_{mPa}$ Filter

The Williams mPa algorithm estimates the cross correlation statistics of the desired and filter input noise components and then removes the biasing effects during the weight update. The advantage of the mPa is that it allows the use of a single input signal and thus a single sensor. In this case, the input and desired signal are the same (13). While most of the $TSAF_{AS}$ filter algorithms are still valid for the $TSAF_{mPa}$, changes were required to the filter update algorithm to account for the use of a single input. In addition, an algorithm was required to generate the statistics for the noise P-vector or P_n. This section first presents the $TSAF_{mPa}$ filter update algorithm which is derived in Appendix D, followed by a description of the

P_n vector estimator. The final portion of this section will overview the $TSAF_{mPa}$ filter implementation.

3.4.1 $TSAF_{mPa}$ Filter Update. The $TSAF_{mPa}$ filter update equation is

$$W_{j+1,k} = W_{j,k} + 2\mu_k \epsilon_k X_{j,k} - 2\mu_k P_{n,j} \tag{3.8}$$

where $P_{n,j} = [p_0\ p_1 ... p_L]_j^T$ and j is the vector index and k is the time index (13:106). The P_n vector is not a function of time as the noise cross correlation statistics are assumed to be stationary, therefore, no time index is required. The $TSAF_{mPa}$ filter with the P_n estimator is shown in Figure 3.2. One thing to note is that the $TSAF_{mPa}$ filter has an additional input which is the P_n vector. In addition, the input and desired signal are one in the same for the single sensor configuration. The

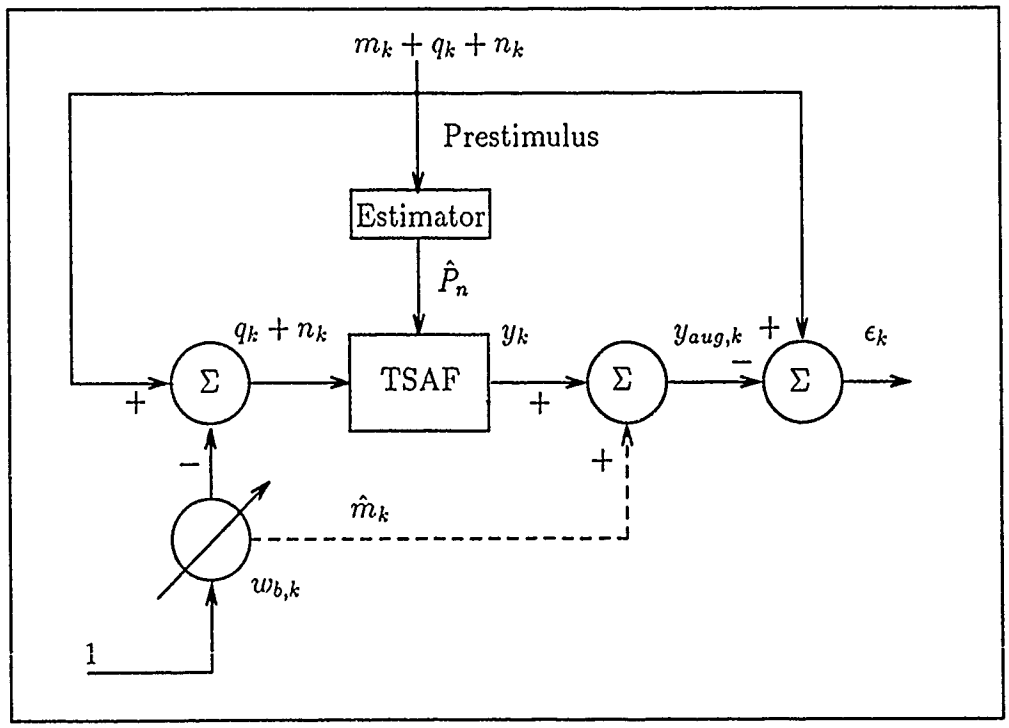

Figure 3.2. $TSAF_{mPa}$ Filter With Single Sensor

role of the P_n vector is to remove the biasing effects of the noise cross correlation statistics from the weight update algorithm (13:97). These statistics are estimated from the pre-stimulus noise. The algorithm for the P_n vector estimator will follow shortly. The next step is to add the bias weight into Equation 3.8 resulting in the following:

$$W_{aug,j+1,k} = \begin{bmatrix} w_{b,k} \\ W_k \end{bmatrix}_j + 2\mu_k \epsilon_k \begin{bmatrix} 1 \\ X_k \end{bmatrix}_j - \begin{bmatrix} 0 \\ P_n \end{bmatrix}_j \tag{3.9}$$

With the change incorporated into the LMS update algorithm, the algorithm to determine the P_n vector is presented next.

3.4.2 P_n Vector Estimator. The P_n estimator is an adaptive filter with the change that each weight is updated independently of the other weights. Figure 3.3 shows a block diagram of a non-causal three tap estimator. The important point is that the estimator must have at least as many taps as the $TSAF_{mPa}$ filter because the P_n vector is directly subtracted from the weight vector. The estimator algorithm is

$$P_{n,j,k+1} = P_{n,j,k} + (2)(0.05)(P_{n,j,k+1} - P_{n,j,k}) \tag{3.10}$$

where

$$P_{n,j,k} = [p_0\ p_1 \ldots p_L]^T_{j,k} \tag{3.11}$$

and each individual components of the P vector are defined as

$$p_{i,j,k} = E[n_{d,j,k} n_{x,j,k-i}] \tag{3.12}$$

$n_{d,k}$ is the noise associated with the desired input and $n_{x,k}$ is the noise associated with the input signal. j is the vector index, k is the time index, and $i = 0 \ldots (L)$ and is the offset to sequence the input pre-stimulus noise component. The $TSAF_{mPa}$ estimator estimates the noise cross correlation statistics using the pre-stimulus data and passes the P_n vector estimate to the $TSAF_{mPa}$ for use with the filter update

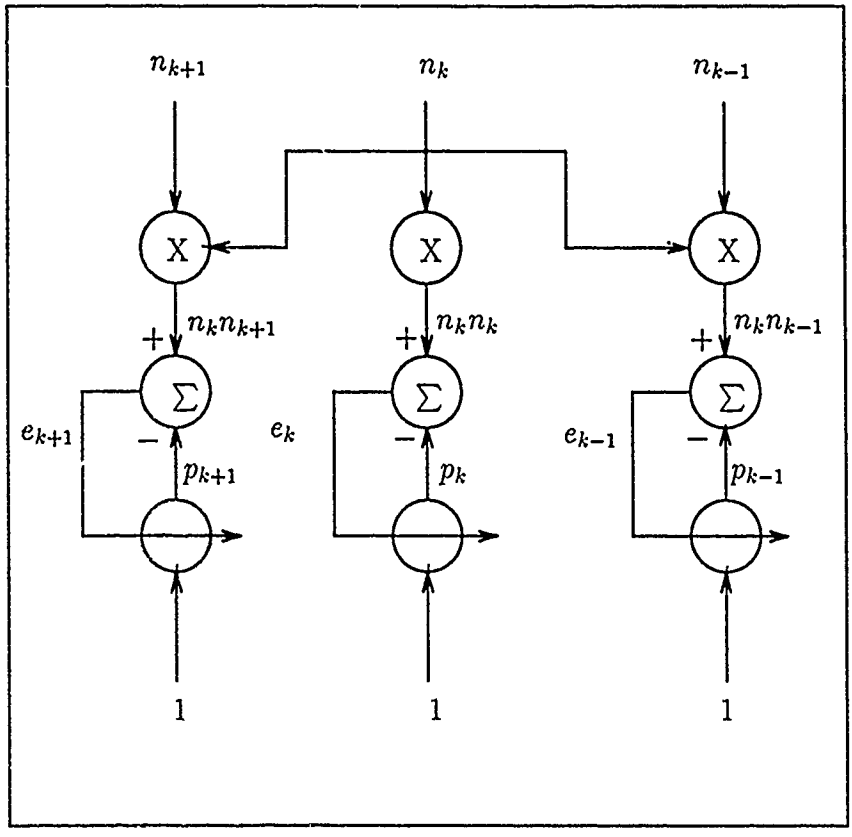

Figure 3.3. Three Tap Non-Causal P_n Vector Estimator (13:93)

equations (13:88). It is important to note that the estimator algorithm is time-sequenced through the pre-stimulus data. However, once the estimator uses all the pre-stimulus data points contained in the input vectors, the taps are frozen and the resulting estimate is passed to the filter. For example, let the input data vector to the filter contain 20 pre- stimulus data points followed by 50 response data points for a total of 70 discrete data points. Again, there are 100 data vectors in all. The P_n estimator works on the first 20 data points and then passes the resulting P_n vector estimate to the TSAF. The $TSAF_{mPa}$ then uses the next 50 data points and the P_n vector to update its filters.

3.4.3 $TSAF_{mPa}$ Software Overview. The $TSAF_{mPa}$ program flow is very similar to that of the $TSAF_{LMS}$. However, there are several additional procedure calls along with changes to existing procedures. The changes are minor and accomplished within the procedure by checking a flag variable. If the flag is set for mPa, then appropriate action is taken within the algorithm execution. Now for the overview of the main program loop which is the following:

- Initialize variables, arrays, and vectors (only once).
- Load pre-stimulus vector.
- Update P_n vector.
- Load input signal.
- Update the bias weight.
- Remove bias from the input signal.
- Calculate and update μ_k.
- Calculate output signal, y_k using mPa.
- Add bias to y_k forming $y_{aug,k}$.
- Calculate error, e_k
- Update weight vector.
- Loop if more data.

Again, Appendix F contains the program listing with a description of the code and major procedures.

3.5 Filter Verification

This section presents the major tests used to verify that the TSAF filter and the individual components were error free. Several tests were performed to debug the initial program and are not listed. The tests presented here are intended to establish the integrity of the software by starting with simple test designed to check individual algorithms and concluding with test which stress all the sub-components of the filter. Therefore, each test builds on subsequent tests until the entire filter is assembled. With this in mind, the tests presented in this chapter are organized as follows:

- Testing of Individual Algorithms. These are the simplest test and are intended to verify the performance of the individual algorithms which the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters share in common. The specific algorithms tested are the following:

  1. μ_k Update
  2. Bias Weight Update
  3. Filter Weight Update

  These test are discussed in more detail in the following sections.

- Testing of the $TSAF_{LMS}$. Once the individual components are verified, the TSAF filter using the LMS algorithm is the next level of complexity. There are two test performed in this section.

  1. $TSAF_{LMS}$ Test I compares experimental results against theoretical results. The input files are noiseless and only contain the jitter component.
  2. $TSAF_{LMS}$ Test II uses human EEG noise added to the noiseless input signal of $TSAF_{LMS}$ Test I.

  The filter performance is characterized by comparing the filter output results from $TSAF_{LMS}$ Test I to those of $TSAF_{LMS}$ Test II.

- Testing of the $TSAF_{mPa}$. The mPa uses the P_n vector to update the filter weights. The P_n vector is generated in a separate algorithm which estimates the cross correlation statistics of the input signals. Therefore, three tests are performed in this section.

    1. The P_n Estimator Test I verifies the performance of the estimator against a single input file which is $AWGN_0$.
    2. The P_n Estimator Test II verifies the estimator taps converge to zero given two uncorrelated input files are used.
    3. The $TSAF_{mPa}$ Test ensures the mPa implementation is error free using the output of the estimator.

3.5.1 Test Data Files. The simulated EP data files were used along with computer generated data files to verify the $TSAF_{LMS}$ and the $TSAF_{mPa}$ filter implementations. In general, the files contained 100 data vectors with 50 discrete sample points in each vector. The reader is referred to Appendix E for further information on the computer generated files. The data files used in this chapter are defined below:

- $AWGN_0$ was additive white gaussian noise with zero mean and unity variance.
- $AWGN_1$ was additive white gaussian noise with zero mean and unity variance and uncorrelated with $AWGN_0$.
- $AWGNBIAS$ was $AWGN_0$ with a time varying bias added.
- QDAT was derived from the simulated EP data set and contained the jitter component, Q_j.
- SDAT contained the mean and jitter components, $M_j + Q_j$, from the simulated EP data set.
- DDAT contained $M_j + Q_j + N_j$ components from the simulated EP data set.

3.5.2 Test of Individual Algorithms. This section performs the following stand alone test of algorithms which are common to the $TSAF_{LMS}$ and the $TSAF_{mPa}$:

- μ_k Update.
- Bias Weight Update.
- Filter Weight Update.

The purpose of the individual cc ıponent testing is to help isolate any deficiencies within the algorithms.

3.5.2.1 μ_k Update Test. This test verified the implementation of the μ_k update algorithm. This algorithm initially had problems in that a signal with zero energy, $E[x_k^2] = 0$, produced an infinite gain constant. This problem was resolved by adding an offset of 0.01 to the calculated signal energy. The test foı this algorithm simply consisted of passing the $AWGN_0$ file through the algorithm. Given the simplicity of the algorithm, this was the only test performed. The algorithm was verified by monitoring the variables during the program execution using the debug features of Turbo Pascal and comparing the values against the calculated values. All the calculations for two iterations were verified. The gain constant update algorithm performed as expected.

3.5.2.2 Bias Weight Update Test. The bias weight tracks the mean of the input signal at each point in time or for each value of k. Therefore, each of the TSAF filters has an associated bias weight which is updated across the ensemble. The bias weight update algorithm will drive each weight to the ensemble average of its respective column, k, in the data ensemble. A preliminary test using AWGNBIAS file was performed. A total of 100 separate data vectors were passed through the filter and then the values of bias weights were compared to the ensemble average of AWGNBIAS. There was no significant difference.

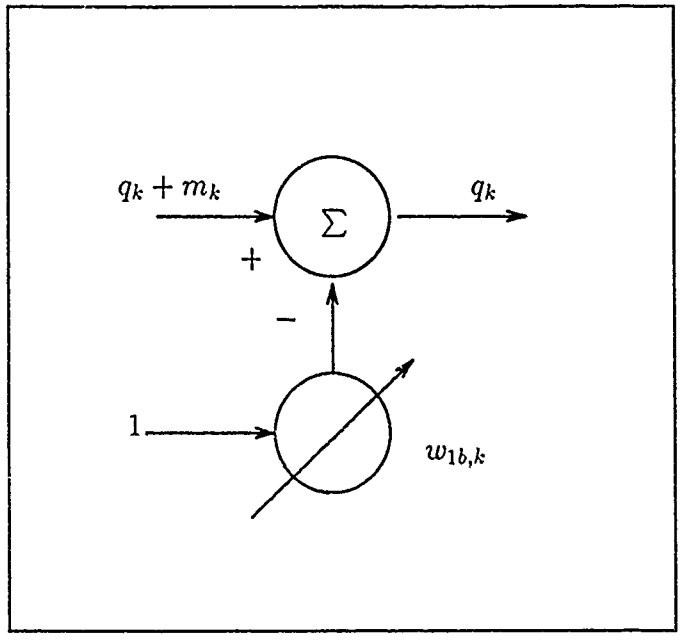

Figure 3.4. Bias Weight Test Configuration

To further test the bias update, the SDAT file was used which contained $m_k + q_k$. The test configuration is shown in Figure 3.4. The expected result was that the bias weights would settle to m_k which was already calculated and discussed in Chapter 2. The test was performed exactly as done for the $AWGN_0$ file. Figure 3.5 shows the actual bias which is m_k. The error $(m_k - w_{b,k})$ is shown in Figure 3.6 and is essentially zero for all the bias weights. Based on these two test, the bias update algorithm was verified and can be assumed to work correctly.

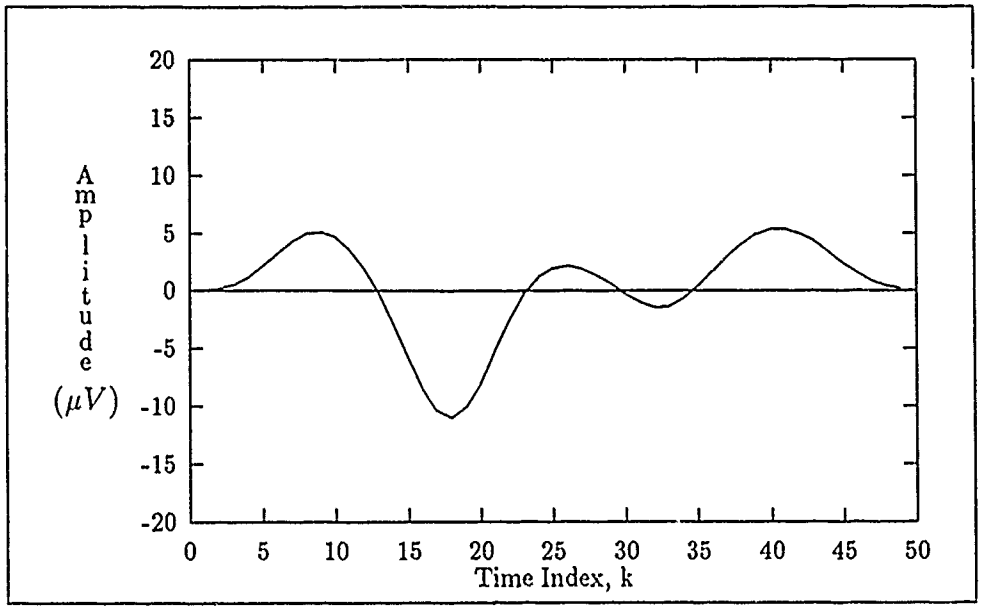

Figure 3.5. Bias Weight Test Expected Results. This plot shows the mean component, m_k, of test signal SDAT.

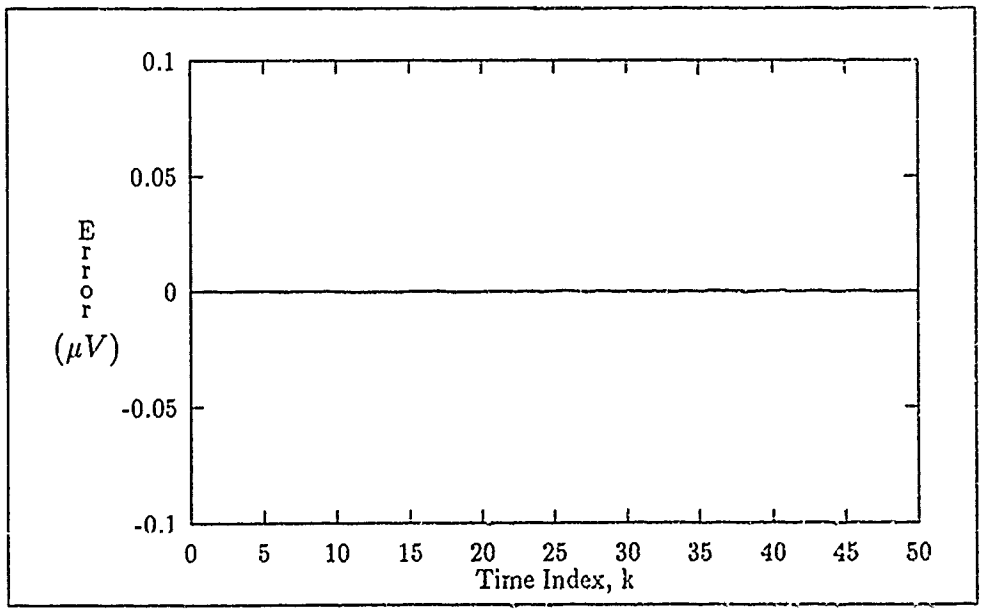

Figure 3.6. Bias Weight Test Error, $m_k - w_{b,k}$

3.5.2.3 Weight Update Test. The previous test confirmed the implementation of the gain constant and bias weight update algorithms. These next test use the TSAF without the bias weight update algorithm. However, the μ_k update algorithm is used to maintain a stable convergence. The following are the two test used to verify the weight update:

- Weight Update Test I. This test used QDAT without noise for both the desired and input signals to test the filters convergence against a theoretical solution.
- Weight Update Test II. Noise is added to the input signal to test the filters ability to uses the correlation between the jitter components to estimate the desired signal.

It is important to note that each test is only valid for the specific settings used in the program initialization. Therefore, both test have a table of the filter settings.

The filter configuration for Weight Update Test I is shown in Figure 3.7. The input signal and the desired signal are the noiseless QDAT file and the filter settings are shown in Table 3.1. The reader is referred to Appendix C for the calculation of the optimum filter solution when the input and desired signal are the same. The calculation is done for both the causal and non-causal case. The test results presented here are for the non-causal filter. Initially the filter did not converge to the theoretical solution shown in Table 3.2. However, the weights were approaching the solution with each pass at the data ensemble indicating the filter was moving slowly towards the desired solution. Given the input and desired signal were both noiseless and exactly the same signal, one would expect the filter to converge rather quickly to the theoretical solution. We will now digress and discuss a change made to the μ_k algorithm to speed up the adaptation.

Figure 3.8 is the circuit model for the μ_k algorithm as defined in Equation 3.7. Note that the $\hat{E}[x_k^2]$ term is an estimate of the signal energy which can be written as

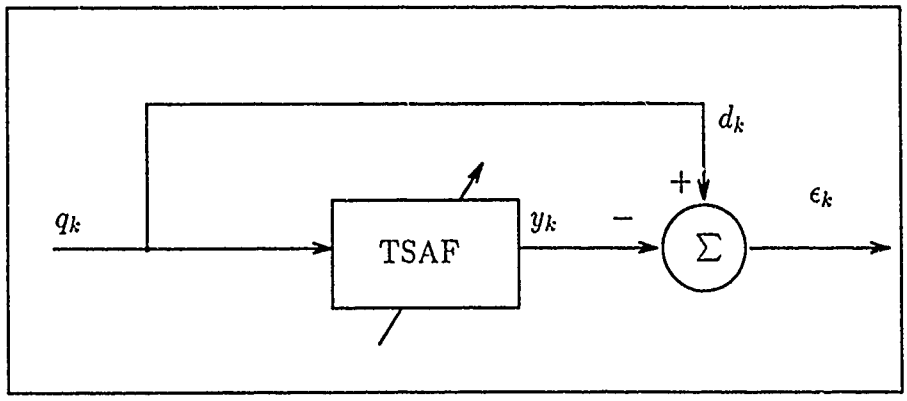

Figure 3.7. Weight Update Test I Configuration. The input signal is the jitter component of the simulate EP data.

Parameter	*Setting*
Number of Runs	20
Number of Taps	3
Misadjustment	0.5
Mode	Non-causal
Algorithm	LMS

Table 3.1. Weight Update Test I Filter Settings

$$\hat{E}[x_k^2] = E[x_k^2] + n_k \tag{3.13}$$

where n_k is the adaptation noise which is passed to the leaky integrator stage and $E[x_k^2]$ is the true signal energy. This noise is captured by the long memory of the integrator and corrupts future calculations of μ_k. The noise present reduces the value of the gain constant and slows the adaptation. To reduce the noise, an estimator could be added to Figure 3.8 which more accurately estimates the signal energy and reduces n_k over time. However, a simpler approach is Figure 3.9 where the integrator stage is dropped and the update for μ_k is instantaneous without memory of past values. Therefore, the μ_k algorithm was modified to the following:

$$\mu_k = \frac{M}{(L+1)(E[x_k^2])(Counter)} \tag{3.14}$$

The *Counter* is incremented at the end of the data ensemble to linearly decreases μ_k as a function of the number of passes the filter makes at the data ensemble. This allows the use of a larger M at the start to rapidly move the weights towards the desired solution. Over time, the *Counter* decreases the gain constant and allows the filter to fine tune itself. All the test presented here use this algorithm and the performance of the filter improved in terms of converging to the desired solution in less time. Table 3.3 shows the experimental results for selected filters with the instantaneous update of μ_k shown in Figure 3.9. The filters converged to the theoretical solution indicating the weight update algorithm was working correctly.

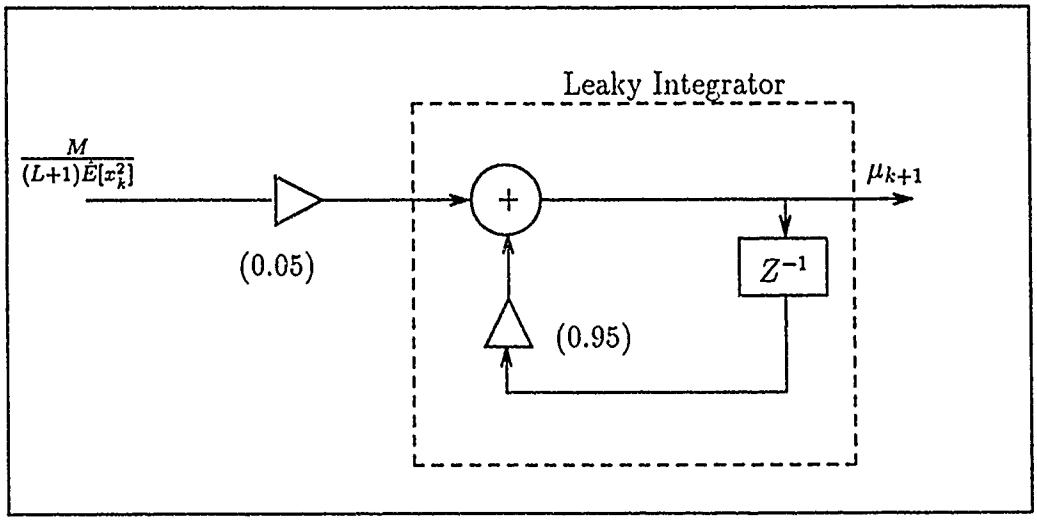

Figure 3.8. Circuit Model for μ_k Update Algorithm

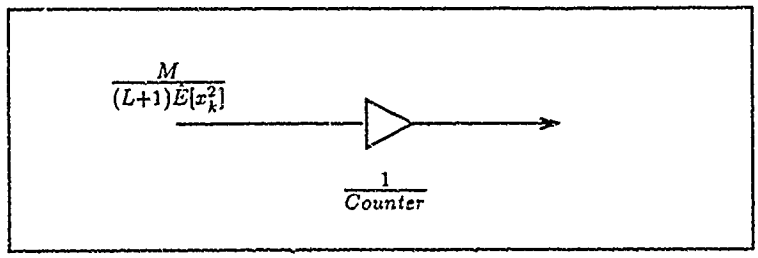

Figure 3.9. Circuit Model of Improved μ_k Update Algorithm

	Filter						
Tap	5	6	20	21	34	35	47
w_1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
w_0	1.000	1.000	1.000	1.000	1.000	1.000	1.000
w_{-1}	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.2. Weight Update Test I Theoretical Solution. The weight values are for a non-causal three tap filter.

	Filter						
Tap	5	6	23	24	34	35	47
w_1	0.037	0.004	0.000	0.000	0.000	0.000	0.019
w_0	0.946	0.993	1.000	1.000	1.000	1.000	0.928
w_{-1}	0.027	0.004	0.000	0.000	0.000	0.000	0.090

Table 3.3. Weight Update Test I Experimental Results. The weight values are for a non-causal three tap filter.

The next test is Weight Update Test II and is designed to stress the weight update algorithm by adding noise to the input signal. The purpose is to ensure the algorithm uses the correlation between the jitter components to estimate the desired signal from a noisy input signal. The filter configuration is shown in Figure 3.10 with the filter settings shown in Table 3.4. The noise added to the input signal was the human EEG noise originally used in the simulated EP data files. The filter size was based on the autocorrelation of Q_j in Figure 2.5 which shows there is significant correlation out to $k = \pm 7$. Therefore, the filter was set with 15 taps to use the correlation.

Figures 3.11, 3.12 and 3.13 compare the filter output to the corresponding desired signal. The vectors selected show a worst, moderate, and best case. The error in the filter output for each of the specified vectors is, in part, due to the adaptation process of the algorithm. Recall that the autocorrelation matrix and cross correlation vector are defined in terms of the expected value. This implies that the weights converge to some average value based on the statistics of the data. Therefore, those input vectors which are closely associated with the average will generate a more accurate output versus those vectors which are far away from the average. However, the performance of the filter is considered acceptable and we are ready to test the fully assembled TSAF.

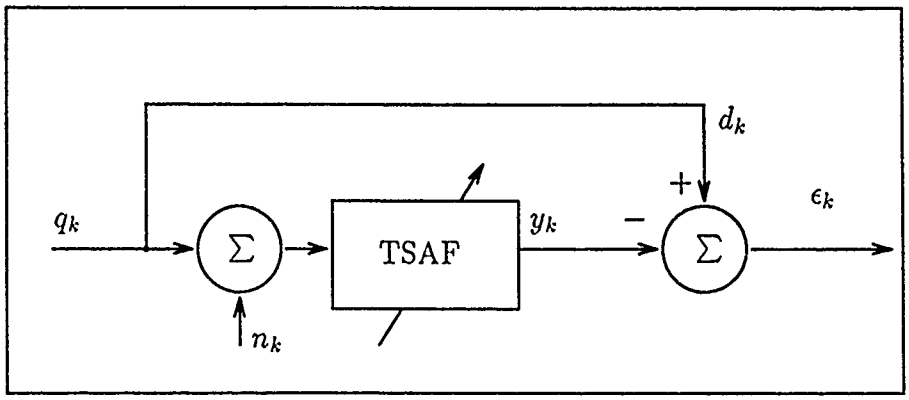

Figure 3.10. Weight Update Test II Configuration

Parameter	*Setting*
Number of Runs	20
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	LMS

Table 3.4. Weight Update Test II Filter Settings.

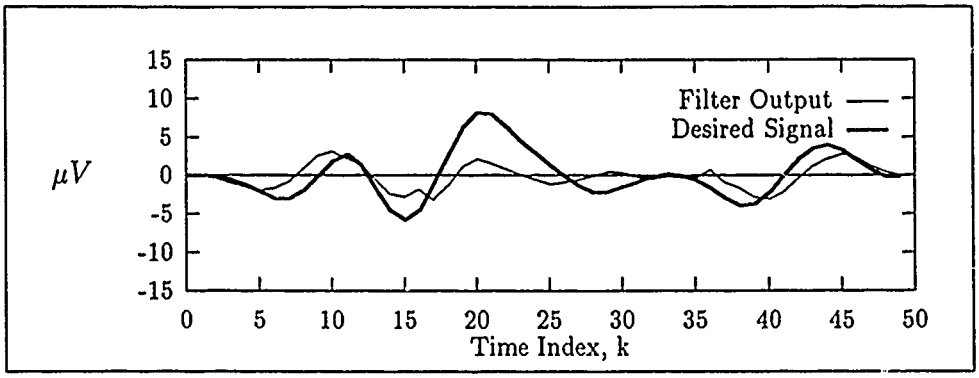

Figure 3.11. Weight Update Test II Results. Comparison of $y_{aug,6}$ and Q_6

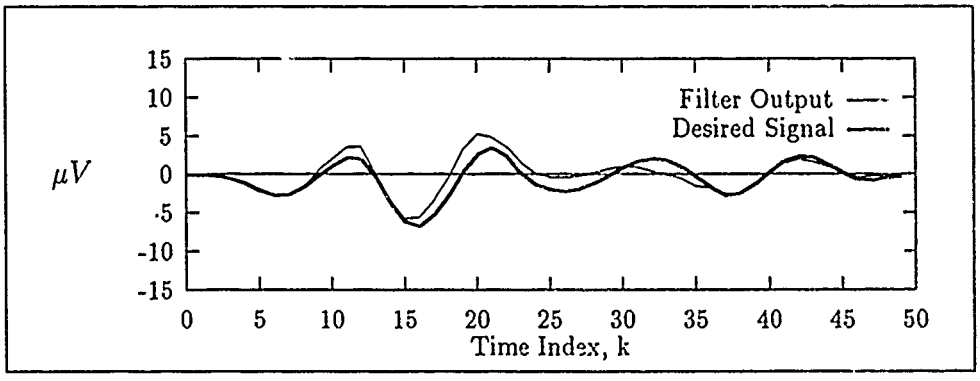

Figure 3.12. Weight Update Test II Results. Comparison of $y_{aug,21}$ and Q_{21}

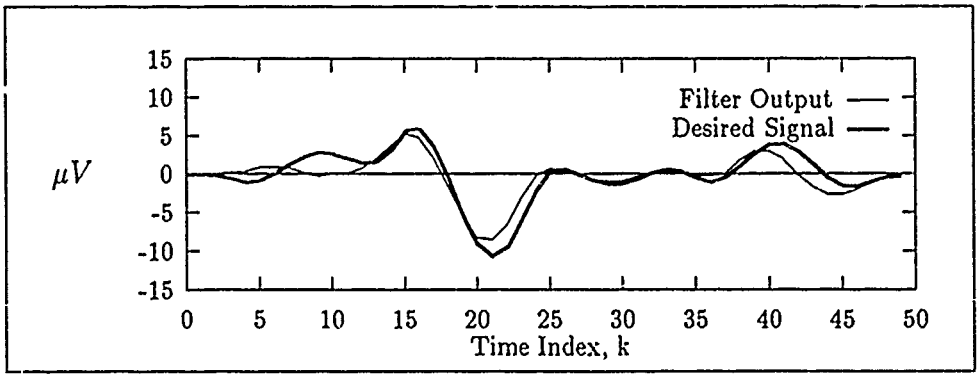

Figure 3.13. Weight Update Test II Results. Comparison of $y_{aug,24}$ and Q_{24}

3.5.3 $TSAF_{LMS}$ Test. With the μ_k update, bias weight, and the weight update algorithms tested and verified, the two test presented here assemble the three sub-components and test the $TSAF_{LMS}$ filter. The two test performed were the following:

- $TSAF_{LMS}$ Test I. This test is similar to the Weight Update Test I in that the filters performance is compared against a theoretical solution. The SDAT file which contains the jitter and mean components is used for the input and desired signal.
- $TSAF_{LMS}$ Test II. This test adds noise to the input signal to verify that the filter removes the mean component of the input signal and then uses the correlation between the jitter components to estimate the desired signal.

3.5.3.1 $TSAF_{LMS}$ Test I. The configuration for $TSAF_{LMS}$ Test I is shown in Figure 3.14 and the filter settings are in Table 3.5. Again, the purpose of this test is to verify that the bias weight removes the mean component of the input signal allowing the filter to use the jitter components of the noiseless input and desired signal to generate an estimate of the desired signal. The expected results from this test, Table 3.6, were exactly the same as for the Weight Update Test I. Given the input and desired signal are the same after the bias weight removes $m_{x,k}$, the center tap should converge to unity with all other taps going to zero.

Initially some of the $TSAF_{LMS}$ filters were diverging from the expected results. In analyzing the filter configuration, it was determined that when the filter initially starts, the output of the bias weight contains a considerable amount of noise due to the small number of samples the bias weight has used to generate an estimate of the true signal mean. To correct this, a delay was placed on starting the TSAF filter adaptation which gave the bias weight adaptation a "head start". The delay was set to 20 which prevents the TSAF from starting until the presentation of vector 21 of

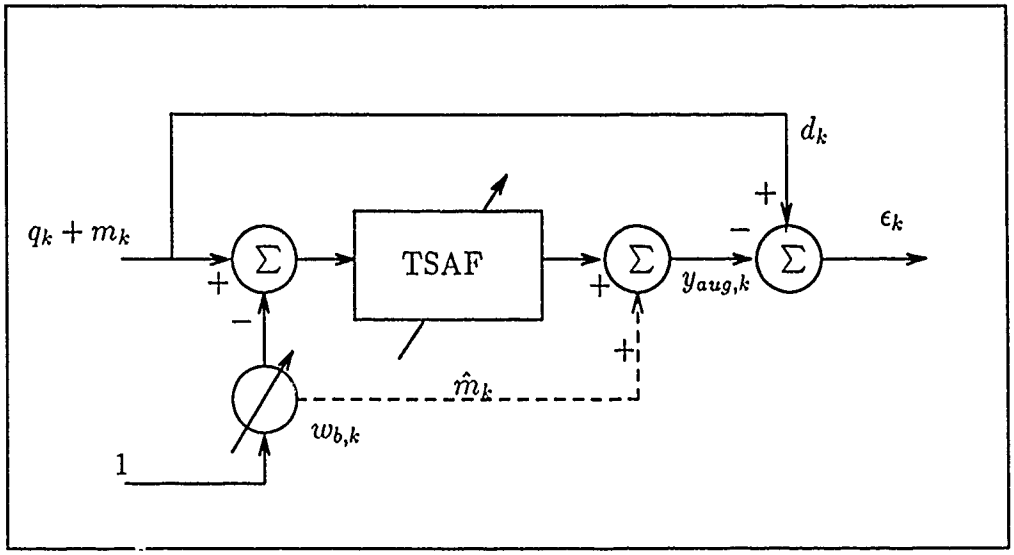

Figure 3.14. $TSAF_{LMS}$ Test I Configuration

Parameter	*Setting*
Number of Runs	20
Number of Taps	15
Misadjustment	0.8
Mode	Non-causal
Algorithm	LMS

Table 3.5. $TSAF_{LMS}$ Test I Filter Settings

the data ensemble. This delay reduced the adaptation noise passed to the filter from the bias weight providing a stable convergence for all of the TSAF filters.

Table 3.7 shows the results from this test and indicates the filters did converge to the desired solution with the filter performing as expected.

	Filter						
Tap	5	6	20	21	34	35	47
w_1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
w_0	1.000	1.000	1.000	1.000	1.000	1.000	1.000
w_{-1}	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3.6. $TSAF_{LMS}$ Test I Theoretical Solution

	Filter						
Tap	1	2	13	24	34	35	47
w_1	0.000	0.007	0.002	0.000	0.000	0.000	0.072
w_0	0.981	0.940	0.997	1.000	1.000	1.000	0.943
w_{-1}	0.155	0.197	0.010	0.000	0.000	0.000	0.015

Table 3.7. $TSAF_{LMS}$ Test I Experimental Results

3.5.3.2 $TSAF_{LMS}$ Test II. The configuration for $TSAF_{LMS}$ Test II is shown in Figure 3.15. The input signal was DDAT and the desired signal was SDAT. The only difference between the two signals was the human EEG noise in the input signal. This test ensures the bias weight removes the mean component of the input signal allowing the $TSAF_{LMS}$ to use the correlation between the jitter components in the presence of noise to generate an estimate of the desired signal. The filter settings are shown in Table 3.8.

The results from the test are shown in Figures 3.16, 3.17,and 3.18 which compares the desired noiseless signal to the filter output. Again, the outputs selected show a worst, moderate, and best case estimation of the signal based on a visual match. The filter output does provide a reasonable estimate of the noiseless desired signal and indicates the filter was performing as expected.

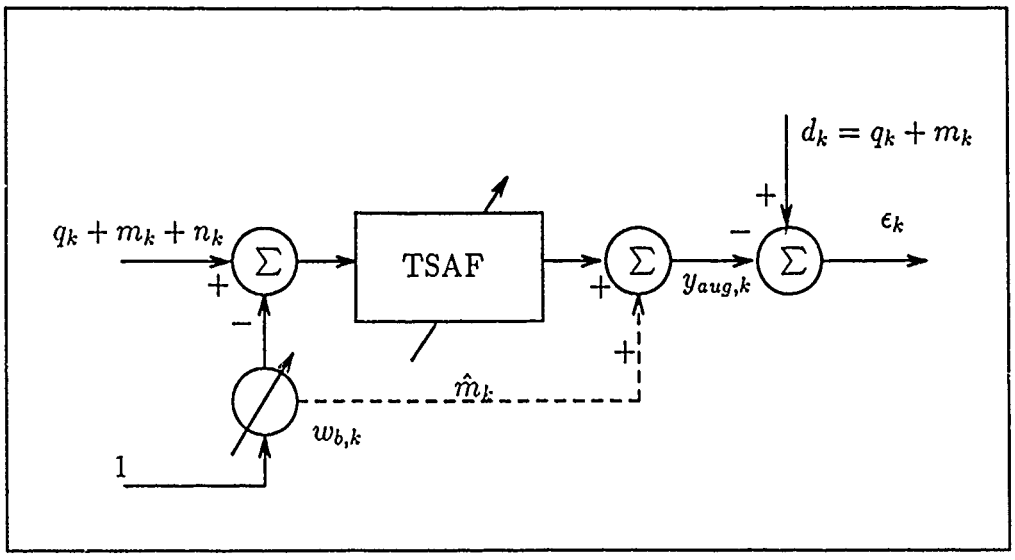

Figure 3.15. $TSAF_{LMS}$ Test II Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	LMS

Table 3.8. $TSAF_{LMS}$ Test II Filter Settings

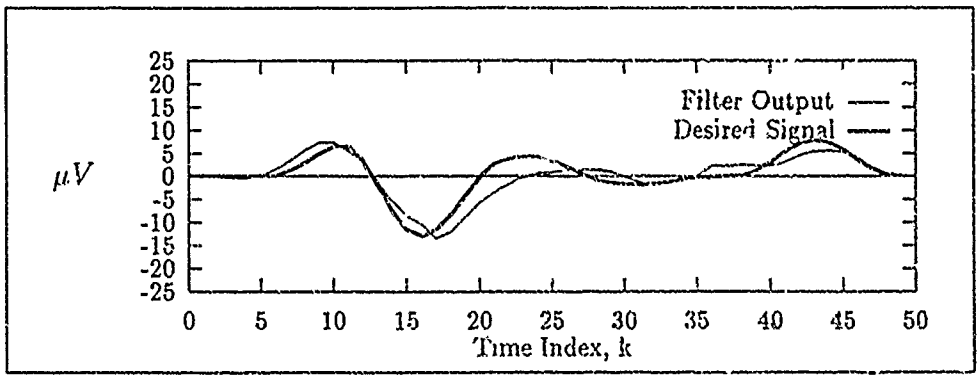

Figure 3.16. $TSAF_{LMS}$ Test II Results. Comparison of $y_{aug,6}$ and Q_6

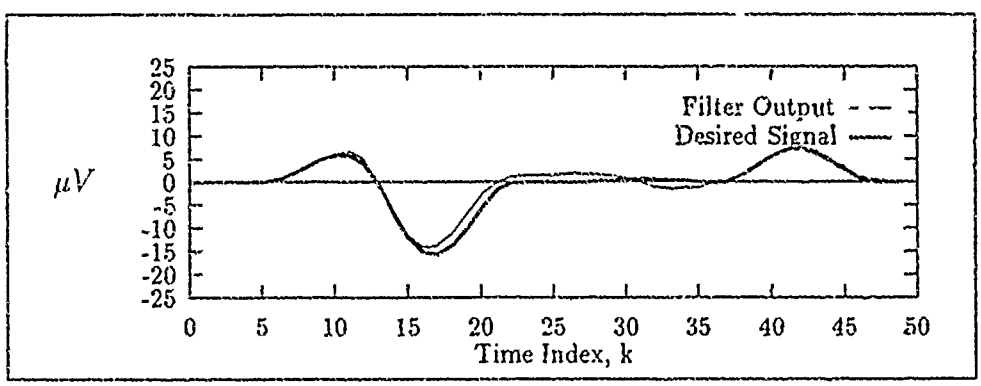

Figure 3.17. $TSAF_{LMS}$ Test II Results. Comparison of $y_{aug,21}$ and Q_{21}

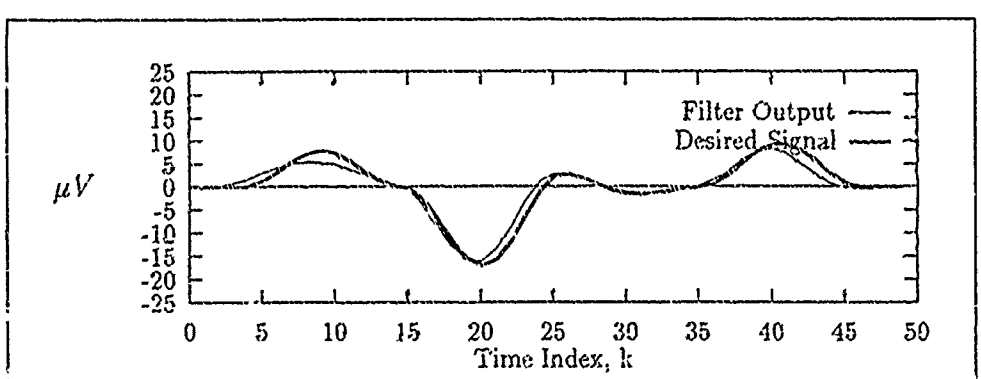

Figure 3.18. $TSAF_{LMS}$ Test II Results. Comparison of $y_{aug,24}$ and Q_{24}

3.5.4 $TSAF_{mPa}$ Verification. Up to this point, there have been five test performed. The first three test verified that the individual algorithms common to the $TSAF_{LMS}$ and $TSAF_{mPa}$ were operating correctly. The last two test verified the performance of the $TSAF_{LMS}$. We are now ready to verify the $TSAF_{mPa}$. The mPa performs two major task. First, it generates an estimate of the cross correlation statistics between the pre-stimulus noise of the input and desired signals and second, the mPa removes the estimate from the weight update. Therefore, the estimator must be verified as well as the mPa weight update algorithm. The following test are performed in this section:

- P_n Estimator Test. These two test are intended to show the estimator is implemented correctly and indeed produces an estimate of the cross correlation statistics of the noise components.
- mPa Test. This test used a computer generated noise file along with QDAT to test the performance of the mPa using the P_n vector generated from the estimator.

3.5.4.1 P_n Estimator Test. The configuration for P_n Estimator Test I is shown in Figure 3.19 where the input noise signal is also the desired noise signal. Given the $AWGN_0$ file was created with zero mean and unity variance, the expected results were that the center tap of the P_n vector would converge to unity with all other taps going to zero. The expected and experimental results are shown in Table 3.9 and confirmed that the estimator performed as expected.

P_n estimator Test II used two separate files for the input and desired signals which were uncorrelated. The test configuration is shown in Figure 3.19. The expected results from this test are that all components of the P_n vector would converge to zero as the files were created with no cross correlation. Table 3.10 shows the expected and experimental results. Again, the estimator performed as expected. Therefore, based on these test the estimator algorithm was verified.

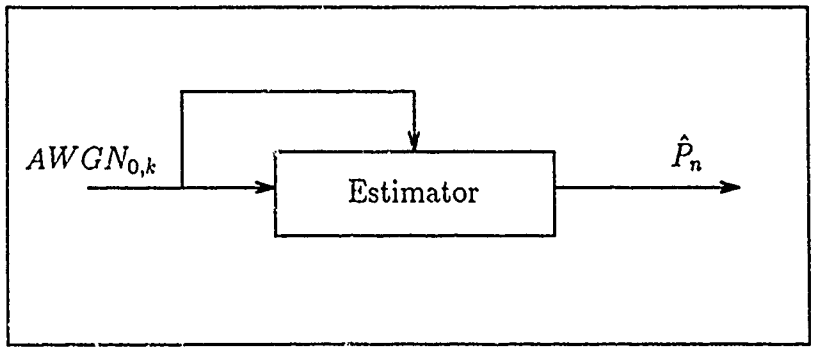

Figure 3.19. P_n Estimator Test I Configuration. The configuration used a 5 tap non-causal estimator resulting in P_n containing 5 points.

Tap	*Expected*	*Experimental*
p_2	0.0000	-0.0191
p_1	0.0000	-0.0136
p_0	1.0000	0.9990
p_{-1}	0.0000	-0.0136
p_{-2}	0.0000	-0.0191

Table 3.9. P_n Estimator Test I Expected and Experimental Results

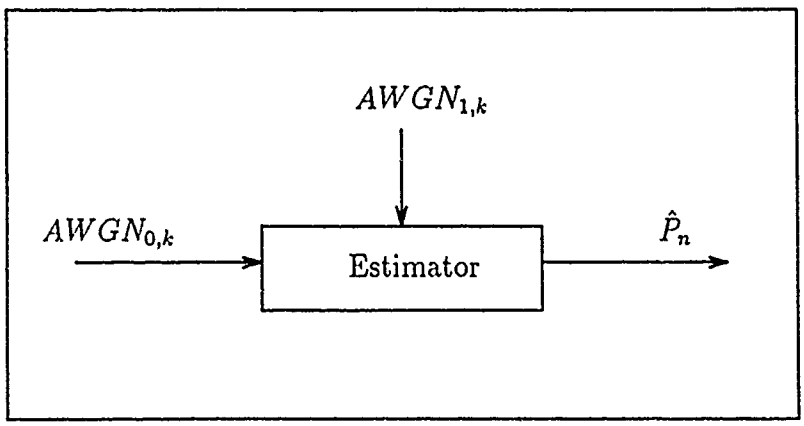

Figure 3.20. P_n Estimator Test II Configuration. The configuration used a 5 tap non-causal estimator resulting in P_n containing 5 points.

Tap	*Expected*	*Experimental*
p_2	0.0000	-0.0024
p_1	0.0000	-0.0031
p_0	0.0000	0.0056
p_{-1}	0.0000	-0.0036
p_{-2}	0.0000	-0.0054

Table 3.10. P_n Estimator Test II Expected and Experimental Results

3.5.4.2 $TSAF_{mPa}$ Test. With the estimator working, the next test verified that the mPa algorithm, using the output from the P_n estimator, was operating correctly. The test configuration is shown in Figure 3.21. The input noise was the NOISE.MPA file which was generated with correlation and contained 100 vectors with 70 points in each vector (see Appendix E). The first 20 samples in each vector provided the pre-stimulus noise. The remaining 50 samples were added to the SDAT file to form the input signal to the filter. Given the filter is using a single sensor (i.e. input and desired signals are the same), the resulting P_n vector contains an estimate of the autocorrelation of the pre-stimulus noise. Table 3.11 shows the filter settings used for this test.

The results are shown in Figures 3.22, 3.23, and 3.24. The filter output does in fact resemble the noiseless desired signal indicating the mPa removed the biasing affects of the correlated noise. Another plot of interest is Figure 3.25 which shows the resulting values in the P_n vector indicating the estimator did detect the correlation in the pre-stimulus noise. This final test verifies the correct operation of the $TSAF_{mPa}$.

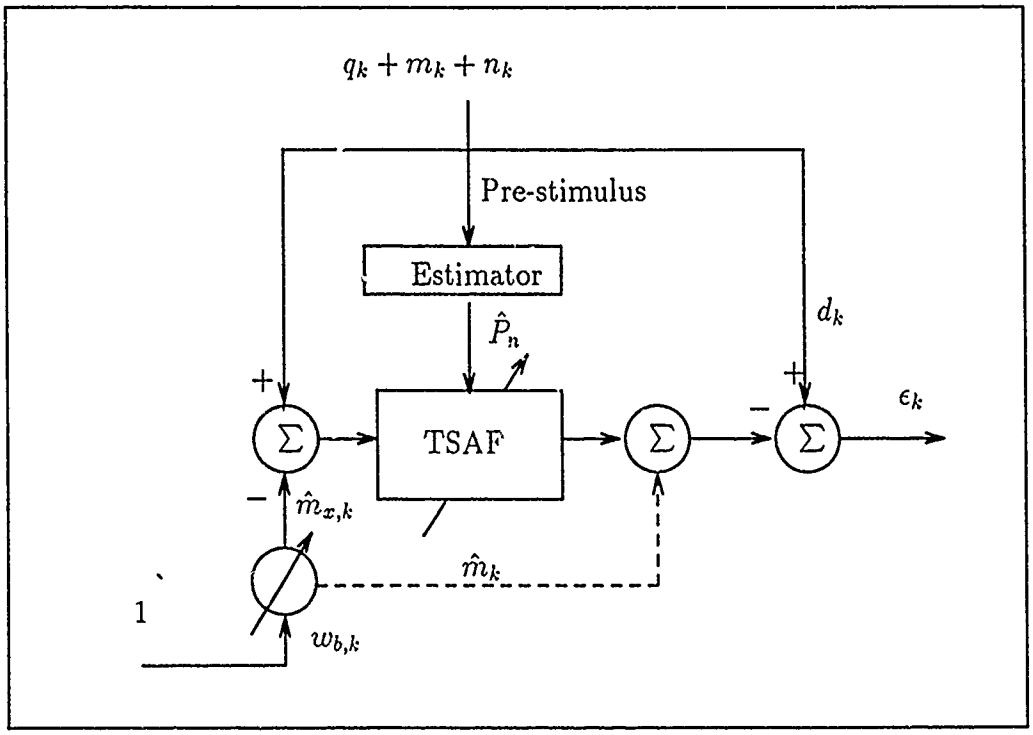

Figure 3.21. $TSAF_{mPa}$ Test Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 3.11. $TSAF_{mPa}$ Test Filter Settings

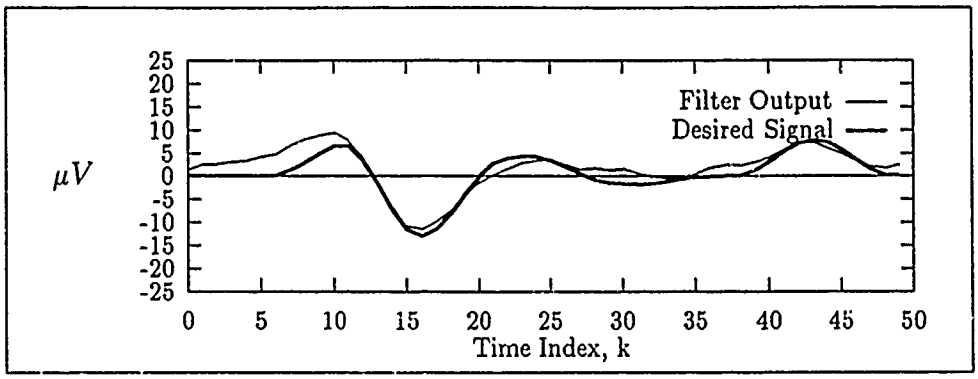

Figure 3.22. $TSAF_{mPa}$ Test Results. Comparison of $y_{aug,6}$ and Q_6

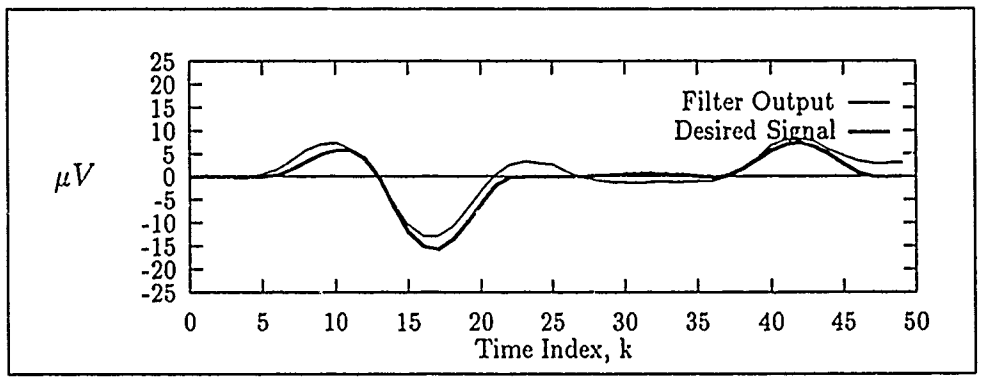

Figure 3.23. $TSAF_{mPa}$ Test Results. Comparison of $y_{aug,21}$ and Q_{21}

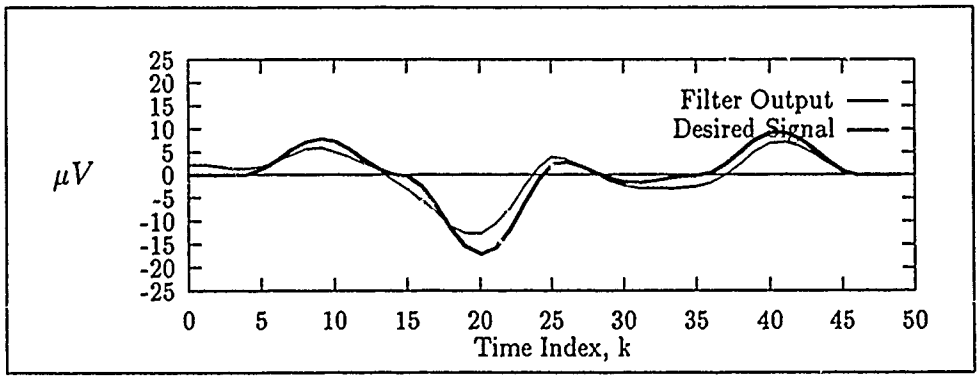

Figure 3.24. $TSAF_{mPa}$ Test Results. Comparison of $y_{aug,24}$ and Q_{24}

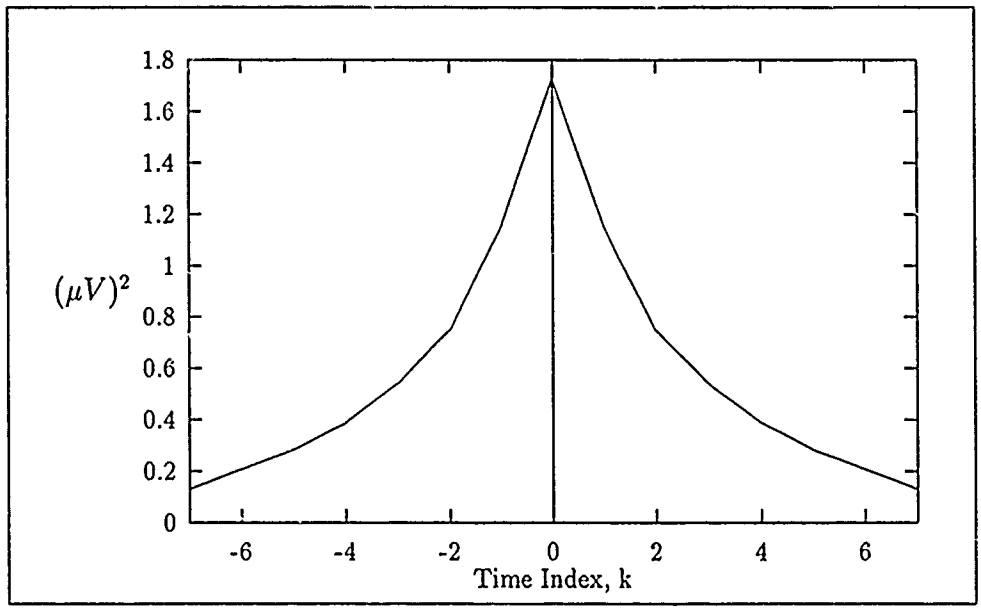

Figure 3.25. $TSAF_{mPa}$ Test Results. Plot of P_n vector.

3.6 Validation of Concept

Up to this point, the $TSAF_{LMS}$ and $TSAF_{mPa}$ have been tested for cases where the filter input signal was noisy while the desired signal was noiseless. However, the test were extremely important in establishing the integrity of the filter implementation. The final test investigates the concept of using human EP to estimate human EF signals.

Testing the concept is accomplished by testing the filter using simulated EF signals created from the simulated EP. The creation of the simulated EF is based on the observation that the ensemble average of the human EP appears highly correlated with the ensemble average of the human EF signal when phase shifted 180 degrees (15:34). An elementary approach to creating simulated EF is to first invert the EP to mimic the 180 degree phase shift and then liner filter the signal. This is discussed in more detail later.

The following presents a layout of this section:

- Analysis of Human EEG and MEG noise. This section performs the cross correlation between the human MEG and EEG noise components as well as the autocorrelation of the MEG. The purpose is to determine if there is any cross correlation between the EEG and MEG based on a limited data set. In addition, the results from the statistical analysis determines how the simulated noise will be generated for the test which follow.
- Concept Test I. This test verifies the filters ability t detect the inversion and forward modeling used to create the simulated EF data file without noise. Using a noiseless input and noiseless desired signal allows us to predict the weight values based on the modeling used to create the simulated EF signal as will be shown.
- Concept Test II. For this test, the $TSAF_{LMS}$ is used with computer generated noise added to the simulated EF and EP signals. The configuration for this

test is identical to the one used in Chapter 4 and is intended to test the ability of the $TSAF_{LMS}$ filter to estimate the simulated EF signal in the presence of noise and resolve the forward modeling use to create the simulated EF.

- Concept Test III. Now the $TSAF_{mPa}$ filter is used with the same data files as in Concept Test II. The purpose of this test is to assess the ability of the $TSAF_{mPa}$ filter to estimate the simulated EF signal in the presence of noise and resolve the forward modeling. The ultimate goal is to compare the LMS performance to the mPa when there is cross correlation between the noise components.
- Comparison of the $TSAF_{LMS}$ and $TSAF_{mPa}$ Performance. This section presents a comparison of the performance between the different test configurations in terms of the averaged squared error. Since these test use deterministic signals, a direct comparison of the filter output to the noiseless desired response is possible.

3.6.1 Analysis of Human EEG and MEG. The LMS algorithm assumes there is no correlation between the noise components and high degree of correlation between the jitter components. Any correlation between the noise components will cause the LMS algorithm to converge to a biased filter solution. Therefore, before jumping directly into testing, it is appropriate to look at the noise statistics.

Human EEG noise and MEG noise were obtained by extracting the pre-stimulus noise from the human EF and EP data files provided by AAMRL. The resulting MEG and EEG files contained 80 vectors with 20 sample points in each vector. A rectangular window was used to perform the cross correlation of the MEG and EEG noise and the autocorrelation of the MEG.

The cross correlation is shown in Figure 3.26 and indicates there is indeed cross correlation between the MEG and EEG noise components in this particular data set. The autocorrelation of the MEG is in Figure 3.27 which shows there is correlation

within the MEG noise itself. It is important to note that this analysis only applies to the specific data files used here and should not be extended to MEG and EEG noise in general without future research. The significance of the cross correlation is two fold:

- First, we expect the $TSAF_{LMS}$ to use the cross correlation between the noise components to bias the weight vector solution. This bias is undesirable, but, unavoidable if the LMS algorithm is used.
- Second, the mPa should see the cross correlation in the noise resulting in non-zero values in the P_n vector. The mPa will then try to reduce the biasing affects by removing an estimate of the cross correlation statistics in the weight update.

Based on these observations, one might expect the $TSAF_{mPa}$ to produce a more accurate estimate of the EF signal over the $TSAF_{LMS}$. To confirm this hypothesis, the noise files used in Concept Test II and III were created with cross correlation to simulate the correlation discovered between the human EEG and MEG. Before testing with correlated noise, the basic filter configuration is tested using noiseless signals.

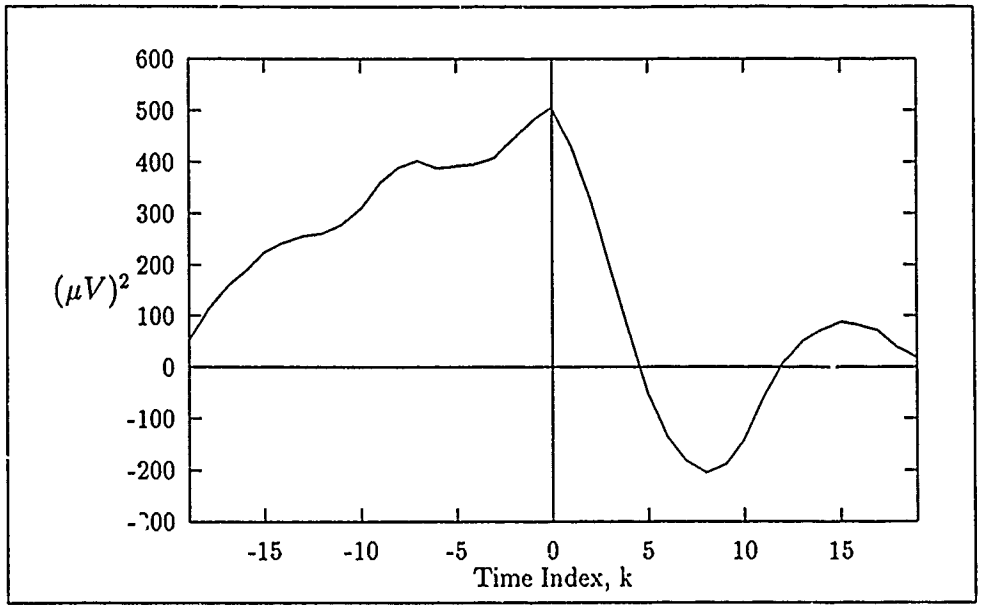

Figure 3.26. Cross Correlation of Human EEF and Human MEG. The correlation was performed using a rectangular window.

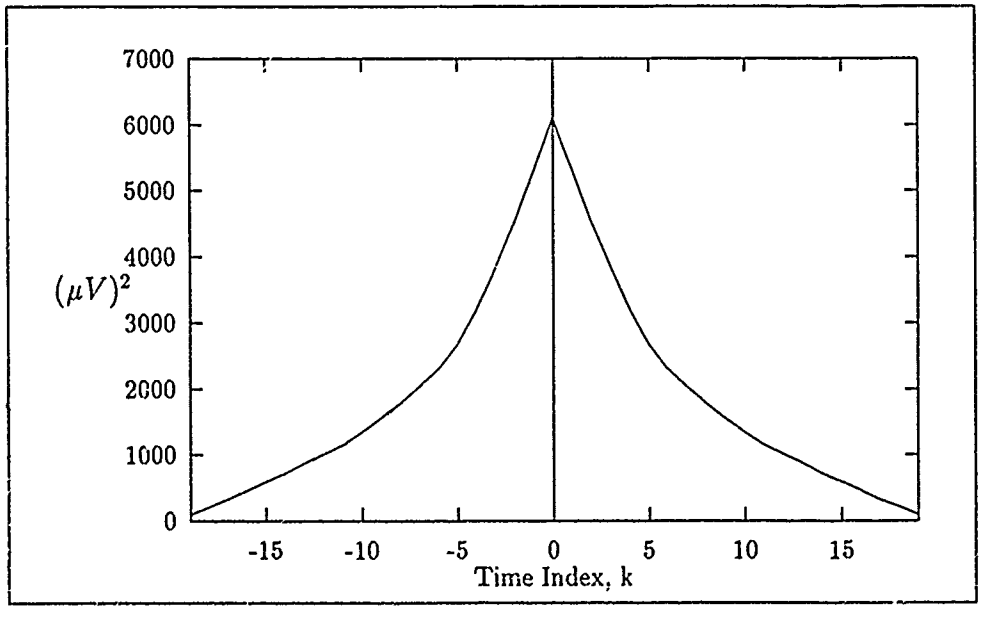

Figure 3.27. Autocorrelation Human MEG. The correlation was performed using a rectangular window.

3.6.2 Concept Test I. The purpose of this preliminary test is to start with a noiseless input and noiseless desired signal and check the filter's ability to estimate the simulated EF using simulated EP. The filter configuration is shown in Figure 3.28 which also shows the forward modeling used to create the simulated EF from simulated EP. The inverter performs the 180 degree phase shift corresponding to the observation that the ensemble average of the human EF appeared highly correlated with the ensemble average of the human EP when phase shifted 180 degrees. The linear filter generates a signal which is correlated with, but, not identical to the simulated EP. Given there is no quantifiable data from which to generate a model, the transfer function of the filter was chosen arbitrarily.

Given the input signals are both noiseless and only differ by the linear operation of the model, the filter weights should converge to model the forward plant and inverter, there by creating an estimate of the simulated EF. In other words, the $TSAF_{LMS}$ must perform the same linear operations on the input signal that were performed to create the desired signal.

It was at this point that the need for an additional bias weight was discovered. The reader may have noticed the additional bias weight, $w_{b,d,k}$, in Figure 3.28. In the two sensor configuration, the mean of the input signal is not necessarily equal to the mean of the desired signal or

$$m_{x,k} \neq m_{d,k} \tag{3.15}$$

The input bias weight only estimates the mean of the input signal and is isolated from the desired signal. Therefore, without $w_{b,d,k}$, the error signal would update the filter weights using the following:

$$\begin{aligned} \epsilon_k &= d_k - y_{aug,k} \\ &= m_{d,k} + q_{d,k} - m_{x,k} - y_k \end{aligned}$$

$$= (m_{d,k} - m_{x,k}) + (q_{d,k} - W_k^T X_k) \tag{3.16}$$

Note the presence of the $(m_{d,k} - m_{x,k})$ term which is the difference between the mean components of input and desired signals. The error term now contains a non-zero mean component, $(m_{d,k} - m_{x,k})$, which the filter can not remove by linearly weighting a zero mean input signal. Therefore, $w_{b,d,k}$ was added to remove the mean component of the desired signal and $m_{x,k}$ is no longer added back to the filter output in the two sensor configuration. This allows the filter to operate on a zero mean input and zero mean desired signal. $m_{d,k}$ is then added back to the filter output as the bias of the desired signal is assumed to be the bias contained in filter output. Given the algorithm was a duplicate of the one used for the input signal, no additional testing was required other than those presented here.

Table 3.12 shows the filter settings used for this test followed by the expected results Table 3.13. The expected results were derived from the forward model used to create the simulated EF. Performing the multiplication indicated by the inverter preceding the filter, the resulting transfer function is

$$0.25 - Z^2 = 0.25Z^0 - Z^2 \tag{3.17}$$

The Z^2 term in Equation 3.17 represents a lag of two which corresponds to filter tap w_{-2} and the Z^0 term corresponds to tap w_0 which is the zero lag tap. The filter weights should converge to the coefficients of the transfer e. . ion which are the values shown in Table 3.13. The test results are shown in Table 3.14 and indicates the filter did indeed converge to a solution which matched the forward modeling.

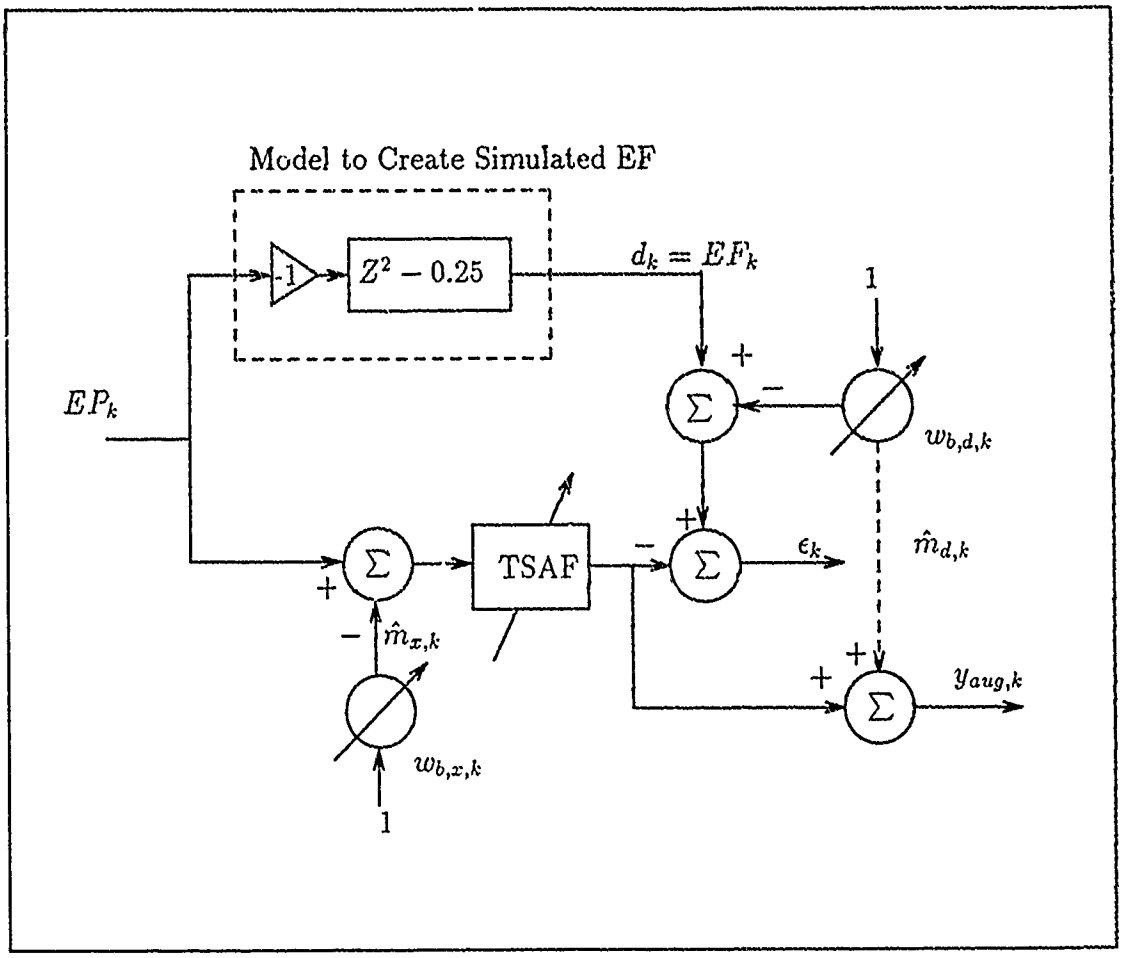

Figure 3.28. Concept Test I Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	5
Misadjustment	0.5
Mode	Non-causal
Algorithm	LMS

Table 3.12. Concept Test I Filter Settings

	Filter			
Tap	10	23	29	43
w_2	0.000	0.000	0.000	0.000
w_1	0.000	0.000	0.000	0.000
w_0	0.250	0.250	0.250	0.250
w_{-1}	0.000	0.000	0.000	0.000
w_{-2}	-1.000	-1.000	-1.000	-1.000

Table 3.13. Concept Test I Expected Results

	Filter			
Tap	10	23	29	43
w_2	-0.064	-0.005	-0.032	-0.097
w_1	0.165	-0.012	0.045	0.184
w_0	0.215	0.287	0.259	0.225
w_{-1}	-0.169	-0.038	-0.054	-0.148
w_{-2}	-0.871	-0.984	-0.968	-0.89387

Table 3.14. Concept Test I Results

3.6.3 Concept Test II The previous test showed that the $TSAF_{LMS}$ was able to resolve the forward modeling used to create the simulated EF for a noiseless input and desired signal. Concept Test II parallels the test to be performed in Chapter 4 in that correlated noise is added to the input and desired signal as shown in Figure 3.29. Recall there was correlation in the human EEG and MEG, therefore, the noise components used in this test were created with cross correlation. The input filter noise file is SIM.EEG and the desired signal noise file is SIM.MEG. Appendix E shows how these noise files were generated.

Table 3.15 shows the filter settings used for this test which also are the same settings to be used in the following chapter. The results from the test are shown in Figures 3.30, 3.31, and 3.32. The plots indicate the filter was generating an output which appears, in some sections, to be offset from the noiseless desired signal. This may be due to the correlation present in the noise components which the LMS algorithm is using to update the filter weights. In effect, the LMS can not differentiate between the jitter cross correlation and the noise cross correlation.

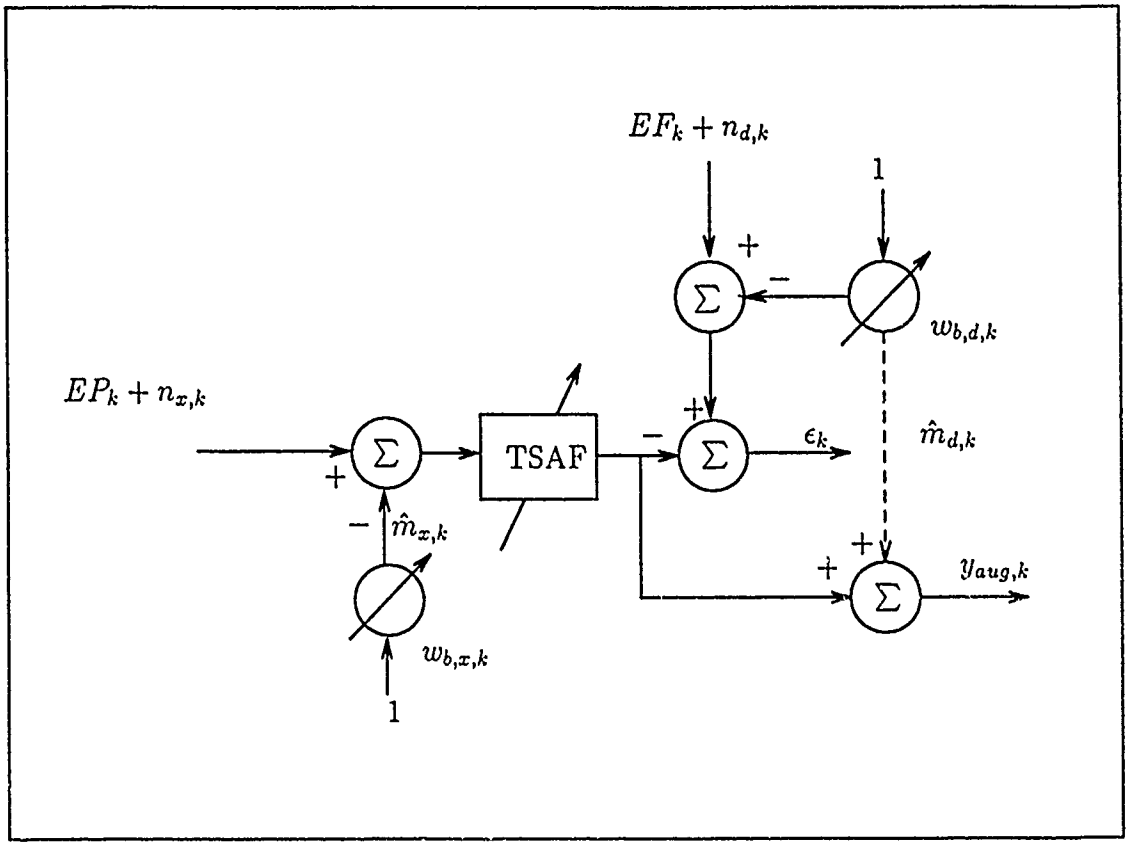

Figure 3.29. Concept Test II Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 3.15. Concept Test II Filter Settings

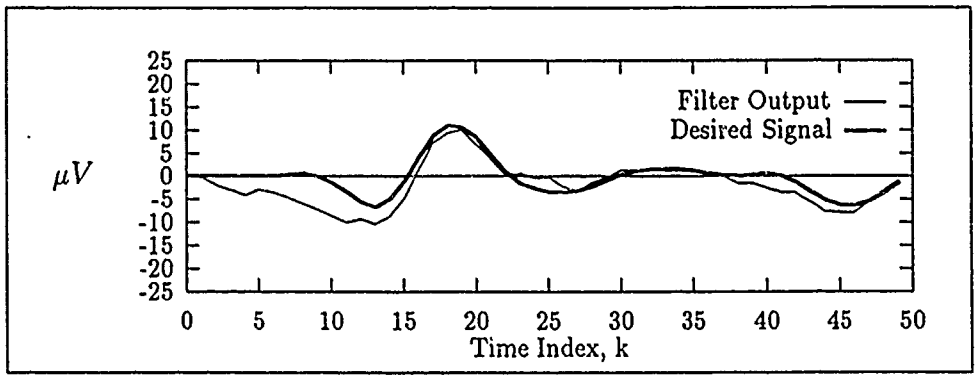

Figure 3.30. Concept Test II Results. Comparison of $y_{aug,6}$ and Q_6

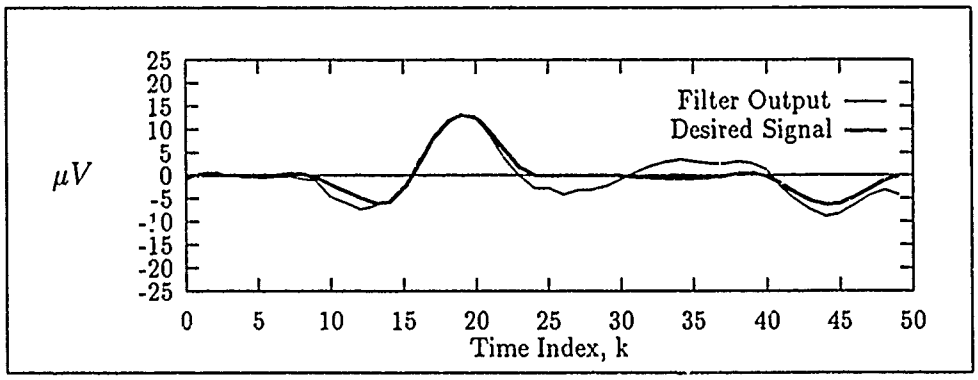

Figure 3.31. Concept Test II Results. Comparison of $y_{aug,21}$ and Q_{21}.

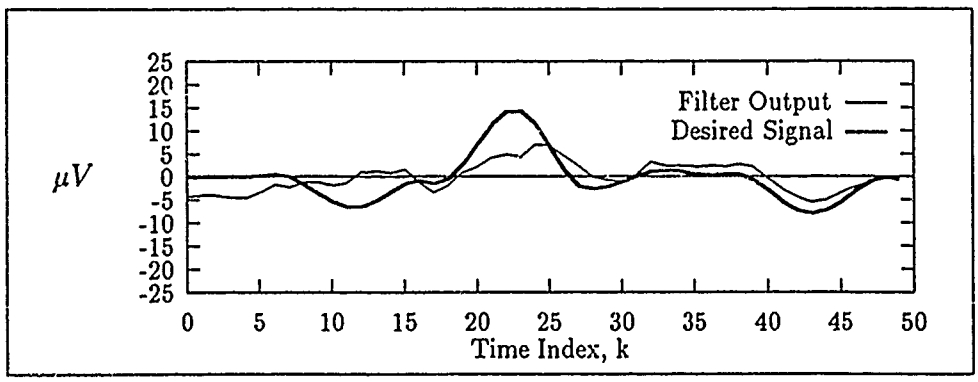

Figure 3.32. Concept Test II Results. Comparison of $y_{aug,24}$ and Q_{24}.

3.6.4 Concept Test III. The purpose of this test is to use the same input signals as in Concept Test II with the $TSAF_{mPa}$ instead of the $TSAF_{LMS}$. The test configuration is shown in Figure 3.32 with the filter settings shown in Table 3.16. Again the configuration and filter settings are identical those to be used in Chapter 4.

The results from this test are shown in Figures 3.34, 3.35, and 3.36 and show a significant improvement over those from Concept Test II which used the $TSAF_{LMS}$. The final data presented for this test is the P_n vector in Figure 3.37 which shows that the estimator detected the cross correlation present in the pre-stimulus noise. One expects that the $TSAF_{mPa}$ used the P_n vector to reduce the biasing effects of the correlation in the filter solution.

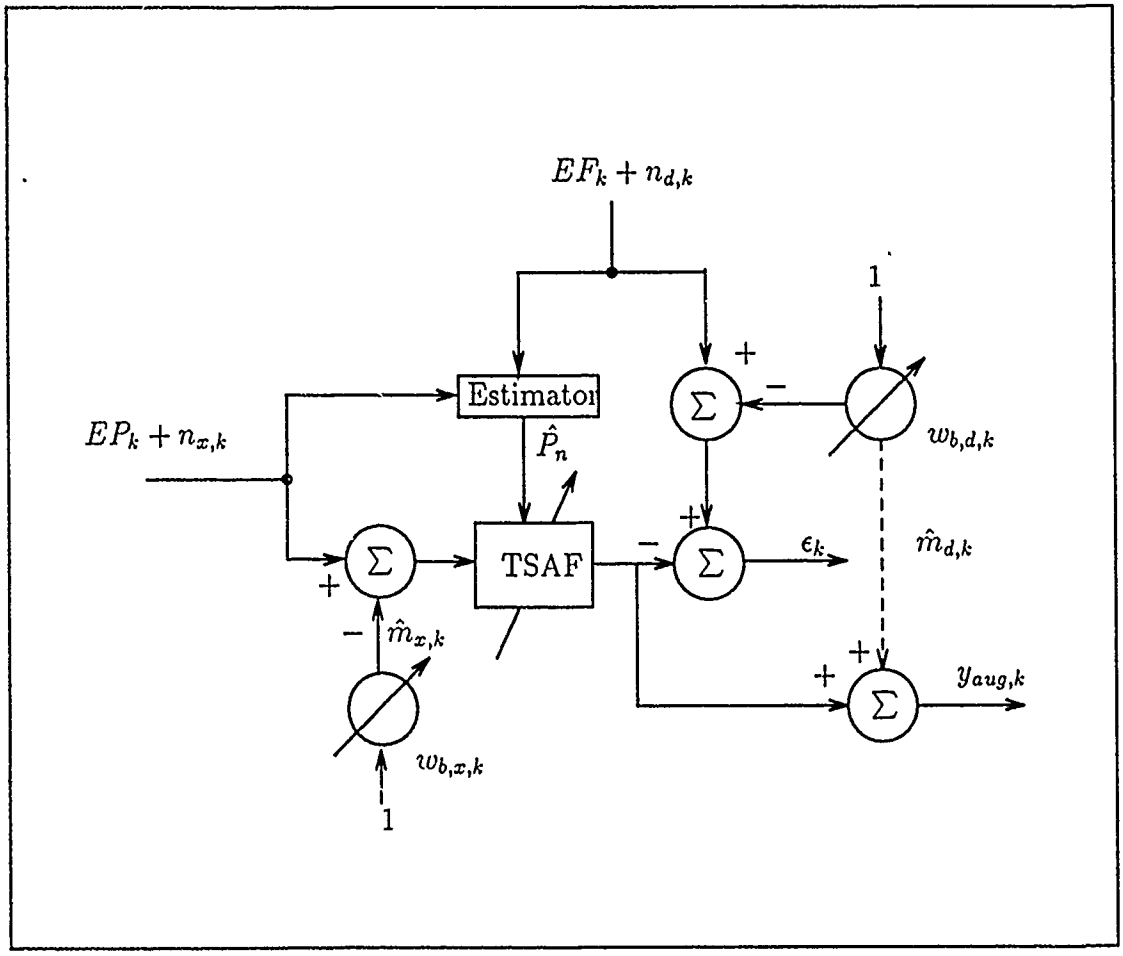

Figure 3.33. Concept Test III Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 3.16. Concept Test III Filter Settings

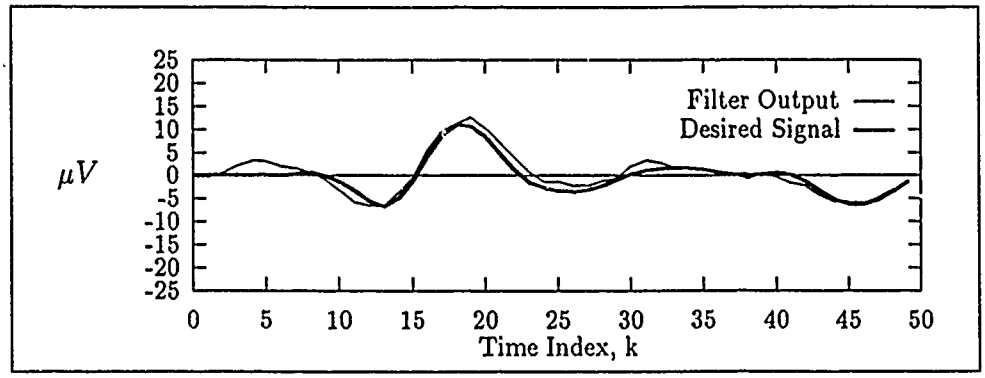

Figure 3.34. Concept Test III Results. Comparison of $y_{aug,6}$ and Q_6

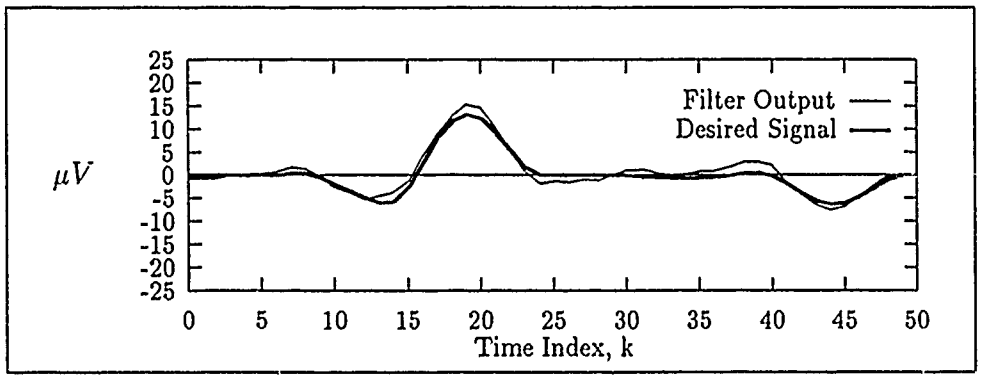

Figure 3.35. Concept Test III Results. Comparison of $y_{aug,21}$ and Q_{21}.

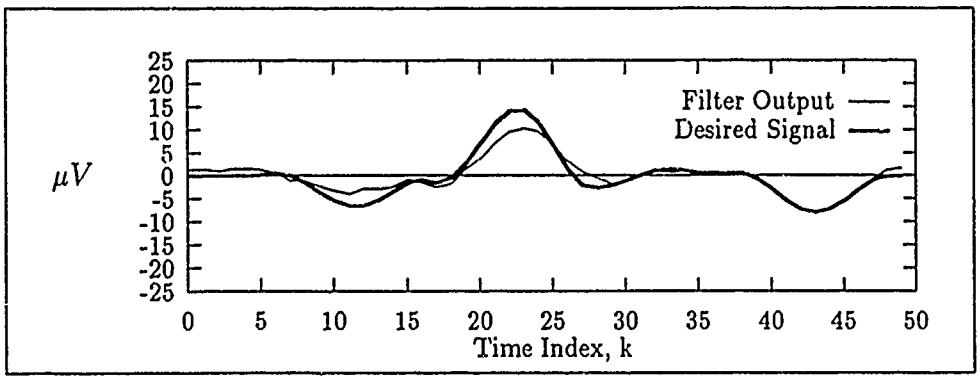

Figure 3.36. Concept Test III Results. Comparison of $y_{aug,24}$ and Q_{24}.

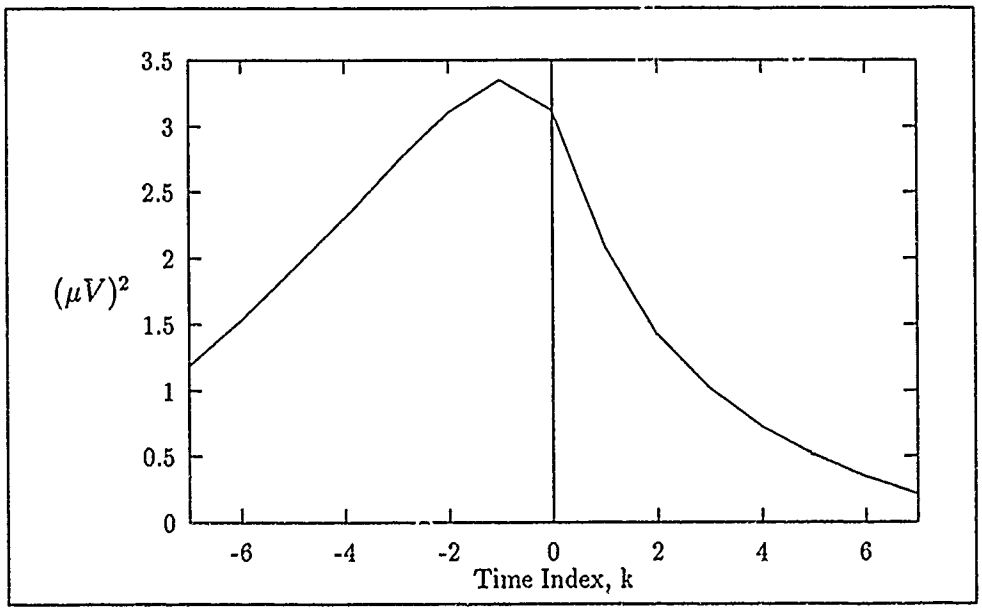

Figure 3.37. Concept Test III Results: P_n Vector which contains an estimate of the cross correlation statistics of the pre-stimulus noise.

3.6.5 Comparison of the $TSAF_{LMS}$ and $TSAF_{mPa}$ Performance. While the filter output from Concept Test III appeared better than the output from Concept Test II, only three of the 80 vectors were displayed. Therefore, the purpose of this section is to quantify the error of all the test used in this section. The figure of merit for a relative comparison between the test is the difference between the filter output, $y_{aug,k}$, and the noiseless desired signal. The error is then squared and averaged over all the data points and vectors. The error for each filter configuration is then defined as the following:

$$\epsilon = \frac{1}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} (SDAT_{j,k} - y_{aug,j,k})^2 \quad . \tag{3.18}$$

where M is the number of data vectors, N is the number of data points in each vector, j is vector index, and k is the time index. Equation 3.18 results is a single number representing an average square error between the noiseless desired signal and the filter output. This calculation is only possible because we used simulated data and, therefore, have a priori knowledge of the true signal component.

For comparison purposes, an additional test was performed with the $TSAF_{mPa}$ using a single senor. The filter configuration was identical to that used in Concept Test III in Figure 3.33 except that both the input and desired signals were the simulated EF buried in the simulated MEG. The filter settings were also the same (see Table 3.16). This test is referred to as Concept Test IV.

The results from the calculation are shown in Table 3.17 It is interesting to note that the $TSAF_{LMS}$ did in fact have an average error which was higher than the $TSAF_{mPa}$. This indicates that the $TSAF_{LMS}$ used the correlation between the noise components to update the weight vector. This resulted in a biased weight vector. On the other hand, the two sensor $TSAF_{mPa}$ appeared to use the cross correlation to generate an estimate of the noise statistics and remove the biasing effects. It is interesting to note that the single sensor $TSAF_{mPa}$ performed better than the two sensor $TSAF_{LMS}$ as well. The average error for Concept Test I is included to

confirm that the filter could resolve the forward modeling for the noiseless signals almost perfectly.

Filter	Error $(\mu V)^2$
Two Sensor $TSAF_{LMS}$: Concept Test II	7.837
Single Sensor $TSAF_{mPa}$: Concept Test IV	4.023
Two Sensor $TSAF_{mPa}$: Concept Test III	2.439
Two Sensor $TSAF_{LMS}$: Concept Test I	0.002

Table 3.17. Comparison of the Average Square Error

3.7 Chapter Summary.

The ultimate goal of this chapter was to test the TSAF in the configurations to be used in Chapter 4 to verify the filter's integrity and validate the concept of estimating human EF with the TSAF. A significant portion of the testing was designed to isolate problems within specific algorithm and/or in the interaction of the sub-components. There were several changes made to the software code to enhance the filter performance. The following presents a summary of these changes:

- μ_k Update Algorithm. This algorithm was modified to use an instantaneous estimate of the signal energy present at the input of the filter. The original algorithm did not perform as well due to the noise present in the estimate of the signal energy. The noise was captured by the long memory of the leaky integrator which then corrupted future updates of the gain constant.
- Filter Delay. A delay was added to the TSAF adaptation to give the bias weight a "head start". This delay reduced the adaptation noise which the bias weight passed to the filter by allowing the bias weight to use the first 20 vectors before the TSAF was started.
- Additional Bias Weight. An additional bias weight was added to remove the mean component of the desired signal. This was necessary because the two stage TSAF removes the input signal mean at the front of the filter and is isolated from the desired signal. The bias weights provide a zero mean input and zero mean desired signal to the two stage TSAF.

To validate the concept of using the TSAF with human signals, a series of test were performed to simulate the filter configurations to be used in Chapter 4. In addition, simulated EF and EP signals were used with computer generated noise. The computer generated noise was created with cross correlation to simulate the cross correlation discovered between the human EEG and human MEG noise components. The results from Concept Test II and III confirmed the hypothesis that the $TSAF_{mPa}$

would generate a more accurate estimate of the desired signal over the $TSAF_{LMS}$ when there is cross correlation between the noise components. It is also interesting to note that the single sensor $TSAF_{mPa}$ had a better average square error than the two sensor $TSAF_{LMS}$ for the specific tests performed in this Chapter.

In summary, the test performed in this chapter were intended to build confidence in the implementation of the $TSAF_{mPa}$ and $TSAF_{LMS}$ filters. Based on the test performed and the results obtained, the $TSAF_{LMS}$ and the $TSAF_{mPa}$ filters were shown to perform as expected with no errors detected in the algorithms. Finally, the concept of estimating human EF using human EP was validated based on the results obtained from using simulated data.

IV. Estimation of Human EF

4.1 Introduction

This chapter summarizes the application of the $TSAF_{mPa}$ and the $TSAF_{LMS}$ filters in estimating human EF. Extensive testing was performed in Chapter 3 to verify that both filters are indeed error free since no standards exist to compare to the filtered human data. In addition, the $TSAF_{LMS}$ and $TSAF_{mPa}$ were tested in configurations identical to the those used in this chapter. By modeling simulated EF using simulated EP, the concept of using EP to estimate EF was validated.

A significant result from the analysis of human EEG and human MEG was that cross correlation was discovered between the noise components for the limited data set. The data used for the analysis is the same data used in this chapter. Based on the results in Chapter 3, one might expect the $TSAF_{mPa}$ to provide a more accurate estimate of the human EF signal. With this in mind, the following is a brief overview of the experiments performed in this chapter:

- Two Sensor $TSAF_{LMS}$ Estimation of Human EF. This experiment uses two sensors to estimate the EF which is equivalent to saying the filter uses two separate input signals. The filter input signal is human EP and EEG contained in the data file EEG.PRN. The desired signal is the human EF and MEG contained in the data file MEG.PRN.
- Two Sensor $TSAF_{mPa}$ Estimation of Human EF. Again, this experiment uses two sensors, but, the mPa is used to estimate the EF signal. The filter input file is EEG.PRN and the desired signal is MEG.PRN.
- Single Sensor $TSAF_{mPa}$ Estimation of Human EF. The mPa is now used with one input file which is the human EF and MEG. The filter input signal is also the desired signal.

Finally, there are two significant assumptions on which the processing in this chapter is based:

1. The statistics of the human EEG and MEG noise are stationary. This means that P_n vector need not be time-sequenced. Recall that the estimator generates the P_n vector based on the statistics of pre-stimulus noise. The resulting P_n vector is then used for all the TSAF filters and is not updated until the next data vector. Therefore, non-stationary noise statistics could degrade the filters performance.
2. There exists correlation between the human EP and EF jitter components. This assumption drives the experiments presented in this chapter in that the correlation between the Q_j components is what the filter uses to update the weights. Ideally, one would like signals with highly correlated jitter components and very little or no cross correlation between the noise components.

4.2. Data Files

There were two data files used in this chapter that were obtained from AAMRL. The MEG.PRN file contained human EF and MEG components and consisted of 80 vectors with 100 discrete sample points in each. Individual vectors contained 0.5 seconds of collection with the first 0.1 seconds being pre-stimulus noise. The sample rate was 200 kHz which means the first 20 sample points were pre-stimulus and the last 80 were post-stimulus. The pre-stimulus was noise only and the post-stimulus contained noise and signal. The EEG.PRN file was in the ne format and contained the human EP and EEG.

In keeping with the signal model presented in Chapter 2, the MEG.PRN file is written as $EF_j + MEG_j = Q_{d,j} + M_{d,j} + N_{d,j}$ where the noise component is MEG and the EF signal component is composed of the jitter and mean. The same model is used for the EEG.PRN data file or $EP_j + EEG_j = Q_{x,j} + M_{x,j} + N_{x,j}$. This is the

notation used to represent the files in the experiment configurations. The x subscript denotes a filter input and the d subscript a desired signal.

4.3 Two Sensor $TSAF_{LMS}$ Estimation Human EF

This experiment estimates the EF signal using the MEG.PRN file as the desired signal and EEG.PRN as the input signal to the $TSAF_{LMS}$. With the LMS algorithm selected, the cross correlation in the MEG and EEG noise components is expected to bias the weight solution as was shown in Chapter 3.

4.3.1 Configuration. The filter configuration is shown in Figure 4.1 with the filter parameters in Table 4.1. The filter configuration in Figure 4.1 is identical to the one used in Concept Test II. Again, the number of taps was based on the filter size used in Chapter 3 to estimate the simulated EP signal buried in human EEG noise. The autocorrelation of the simulate EP signal component showed significant correlation out to $k = \pm 7$. Therefore, the number of taps was set to 15 and the number of runs was 10. The number of runs was based on experienced gained from using the filter and monitoring the instantaneous MSE. For all previous test cases, the filter converged to a solution in less than 10 runs. The filter is said to have converged if the MSE did not appreciably decrease with additional passes through the data ensemble.

4.3.2 Results. The result from this experiment was an output file, YOUT-EXP1.PRN, which contained 80 vectors with 80 discrete points in each. Each of the output vectors represented an estimate of the corresponding desired vector. Six of the output vectors were plotted along with the ensemble average of MEG.PRN in Figures 4.2 through 4.7. The underlying signal component of the human data is not known, therefore, the ensemble average of the MEG.PRN file is provided as a reference to show the deviation of the individual output vectors about the ensemble average. The corresponding input vector is shown as well.

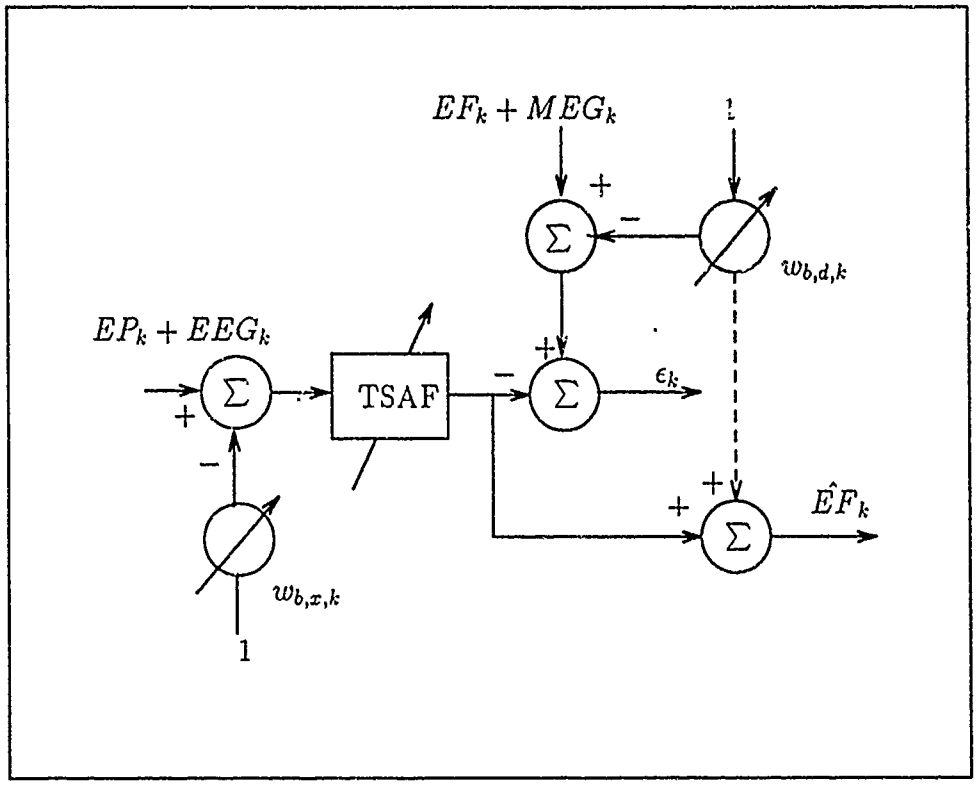

Figure 4.1. Two Sensor $TSAF_{LMS}$ Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	LMS

Table 4.1. Two Sensor $TSAF_{LMS}$ Filter Settings.

Figure 4.8 compares the ensemble average of the desired signal, $E[d_k]$, to that of the filtered output signal, $E[y_{aug,k}]$. The plots are identical which might be surprising at first. However, recall that the two stage filter removes the mean component of the input signal allowing the TSAF to operate on the zero-mean jitter and noise components. Williams points at that "...filtering a zero mean process produces a zero-mean output..."(13:114). The filter output, y_k, is then initially zero-mean until $w_{d,k}$ is added to y_k forming the output signal $y_{aug,k}$. Therefore, the ensemble average of the filter output should equal the ensemble average of the desired signal. Now the filter output signal contains the estimate of the mean component of the desired signal in addition to the filter's estimate of the jitter component. The next experiment uses the mPa to remove the biasing affects of any correlation that might be present between the noise components.

While a qualitative assessment is not possible as we are working with human data, observe in Figure 4.2 that the concept appears to be working in the time interval $k = 15 \ldots 25$. That is to say, the filter output is inverted as compared to the corresponding filter input which agrees with the signal model used in Chapter 3 for the simulated EF.

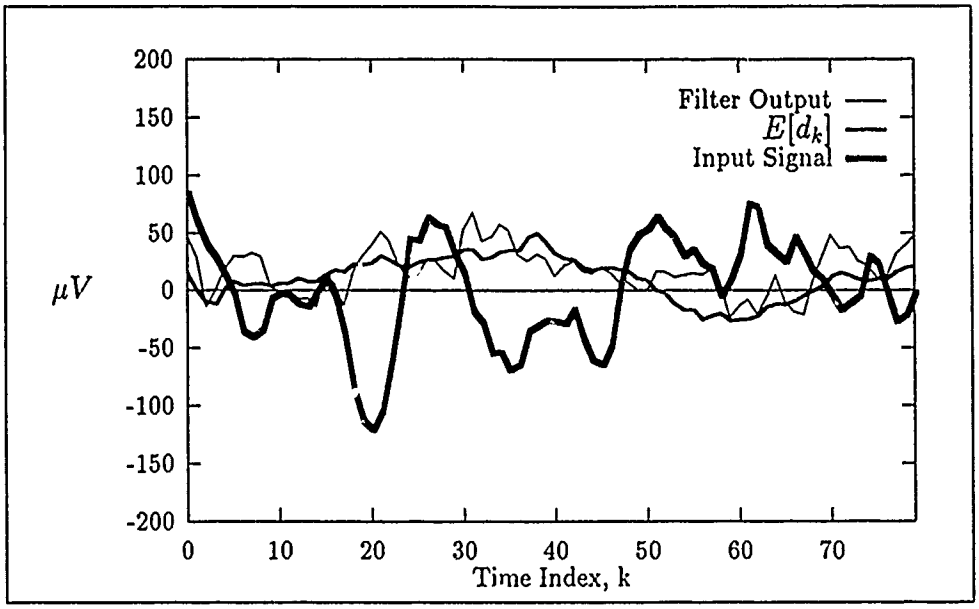

Figure 4.2. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,10}$.

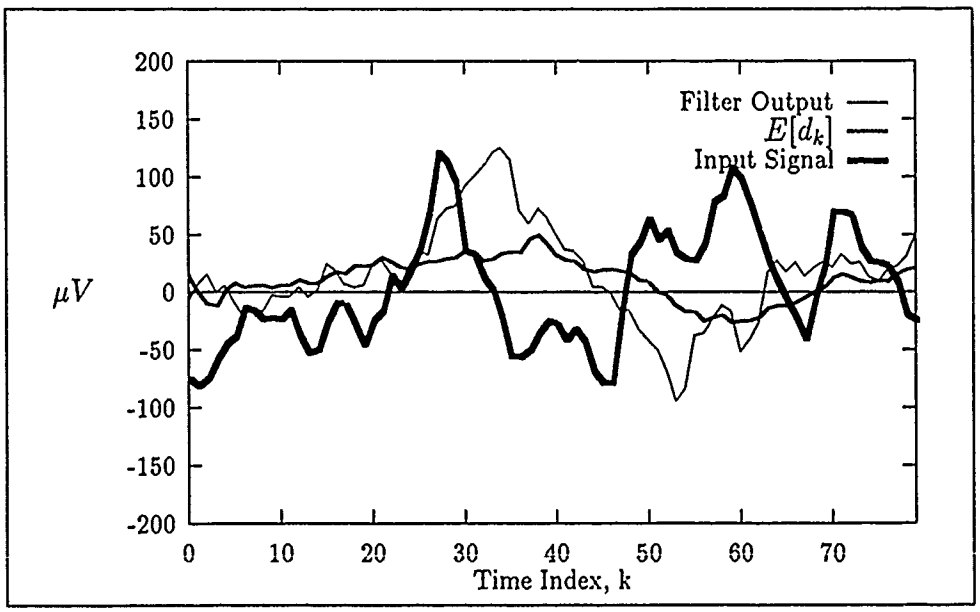

Figure 4.3. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,15}$.

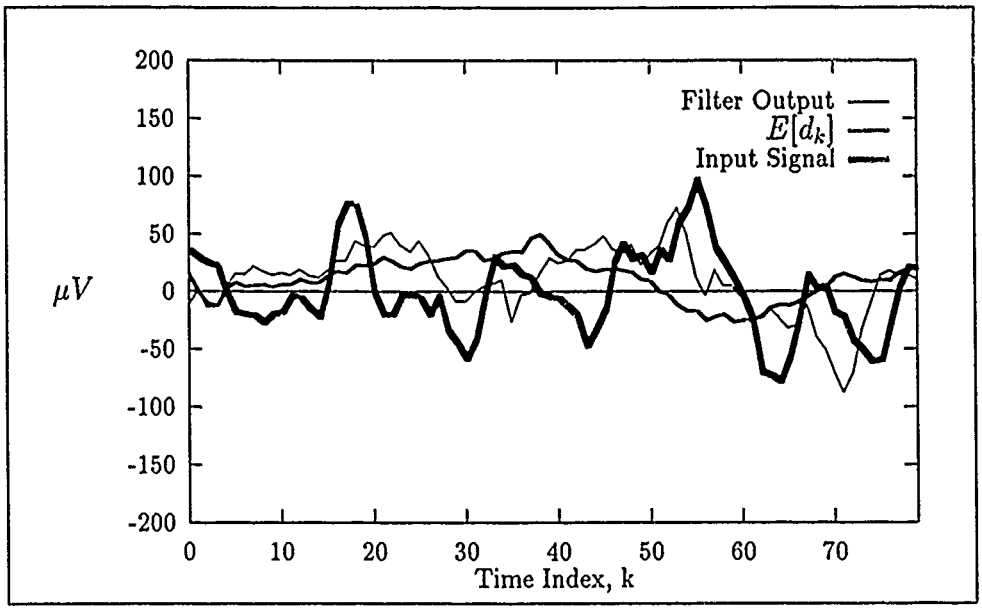

Figure 4.4. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,31}$.

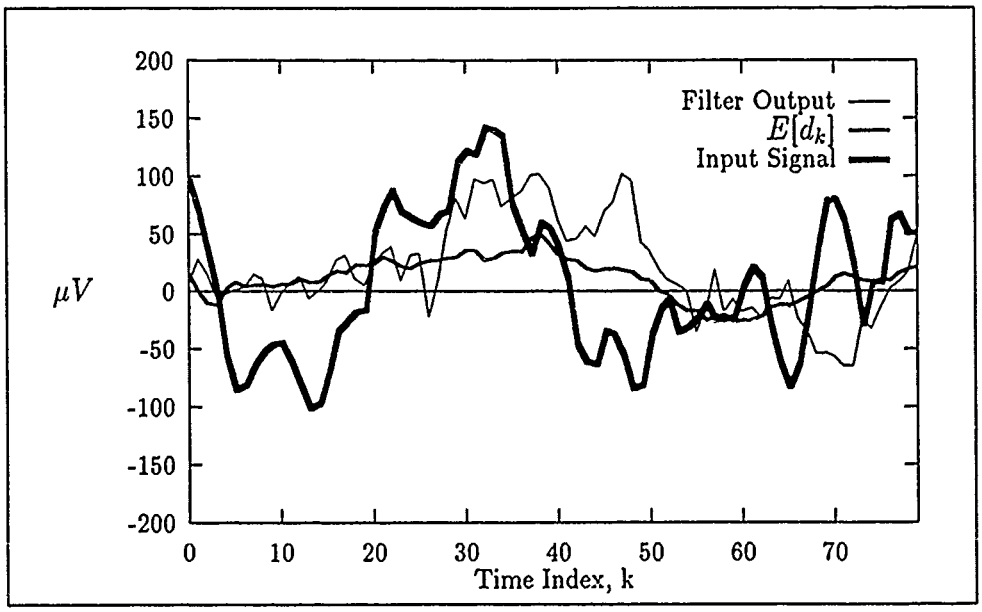

Figure 4.5. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,45}$.

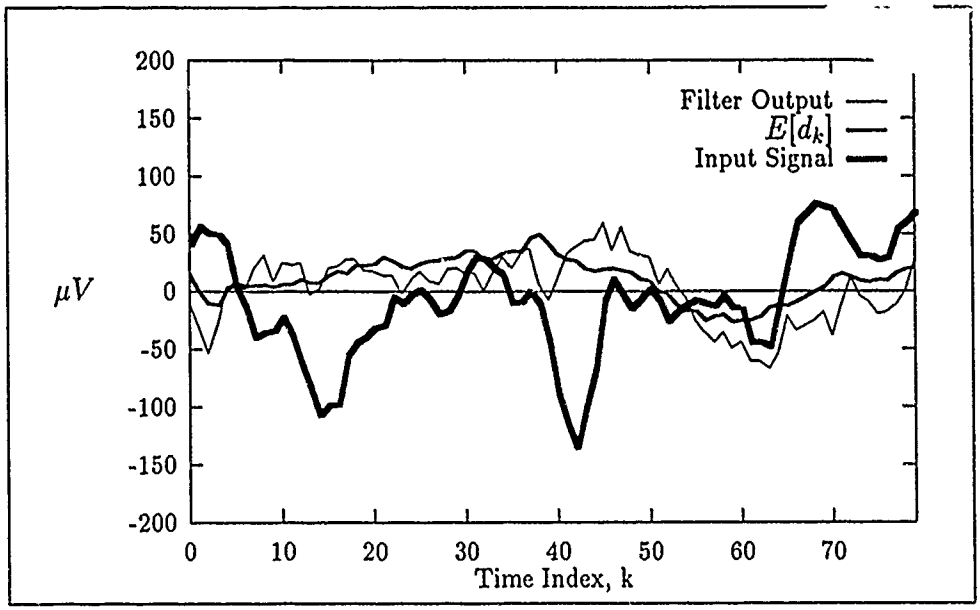

Figure 4.6. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,62}$.

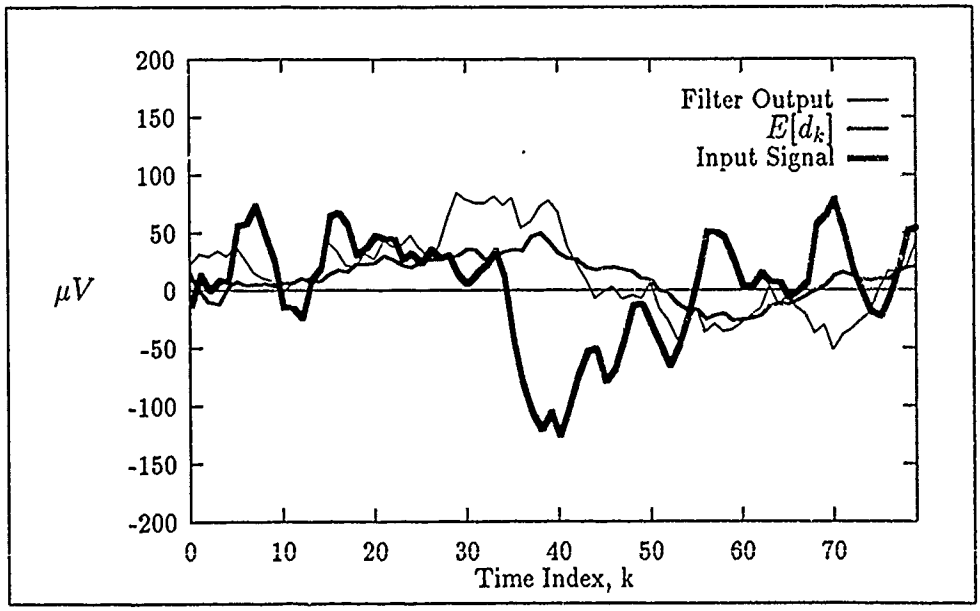

Figure 4.7. Two Sensor $TSAF_{LMS}$ Results: Post-stimulus output vector $y_{aug,70}$.

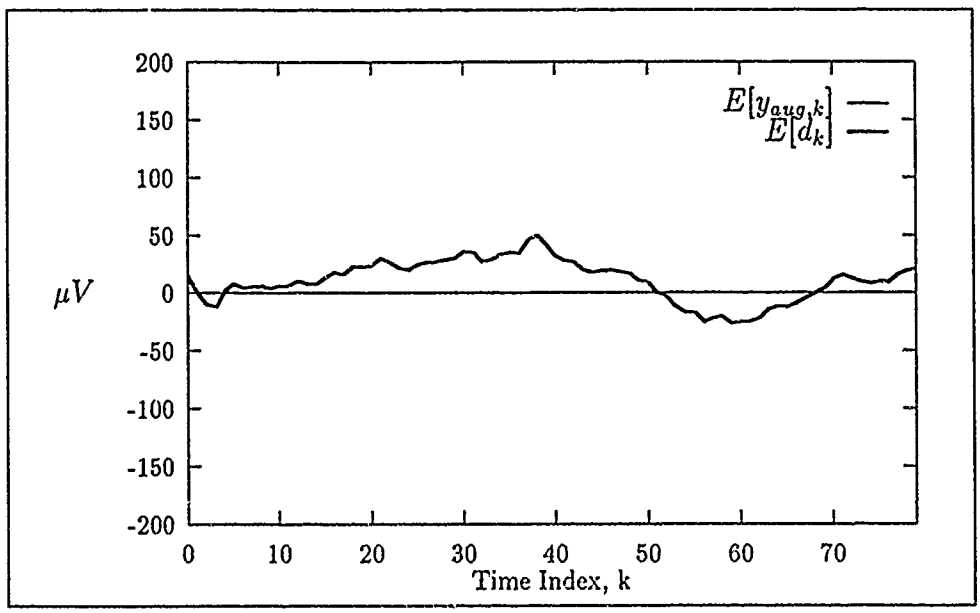

Figure 4.8. Two Sensor $TSAF_{LMS}$ Results: Ensemble Averages of YOUT-EXP1.PRN and MEG.PRN

4.4 Two Sensor $TSAF_{mPa}$ Estimation of Human EF.

This experiment estimates the EF signal component using the TSAF filter in the mPa mode. Based on the test performed in Chapter 3, if there is correlation between the MEG and EEG, the P_n vector will have non-zero components. The assumption is that the noise is stationary which means the P_n vector is time invariant.

4.4.1 Configuration. Figure 4.9 shows the configuration of the filter which is the same as the previous test except now the mPa algorithm is selected. The P_n vector contains the estimate of the cross correlation statistics of the MEG and EEG noise and is passed to the $TSAF_{mPa}$. The estimate is generated from the 20 pre-stimulus data points contained in each data vector.

4.4.2 Results. This experiment produced an output data file, YOUTEXP2.PRN, which contained 80 vectors with 80 discrete points. The output file was the estimate of the EF signal contained in the MEG.PRN. As in the previous experiment, each output vector represents an estimate of the corresponding desired signal vector. Figures 4.10 through 4.15 show the same six vectors as used on Experiment I along with the ensemble average of the MEG.PRN file. Figure 4.16 compares the ensemble average of the filter output file to the ensemble average of MEG.PRN to ensure the filter was not adversely altering the signal. As in the previous experiment, the two plots are identical.

Another interesting plot is Figure 4.17 which shows the values of the P_n vector at the end of the experiment. It appears that the estimator has detected cross correlation between the MEG and the EEG noise components which agrees with the findings in Chapter 3.

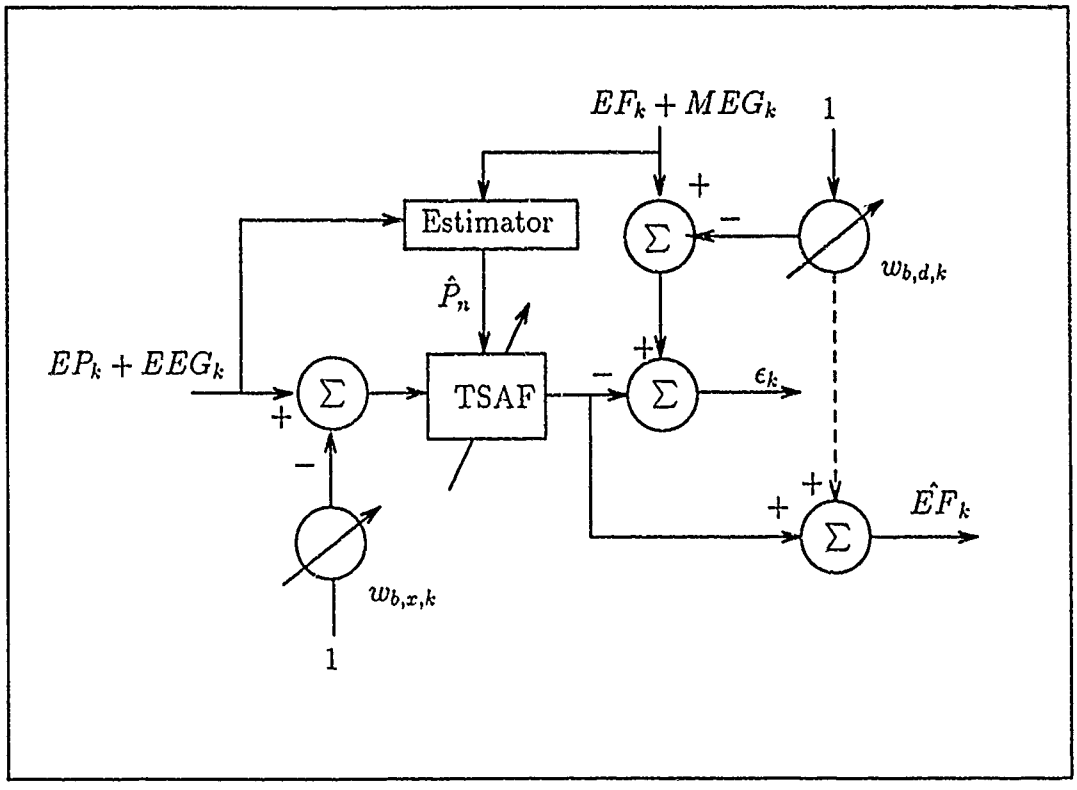

Figure 4.9. Two Sensor $TSAF_{mPa}$ Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 4.2. Two Sensor $TSAF_{mPa}$ Filter Settings

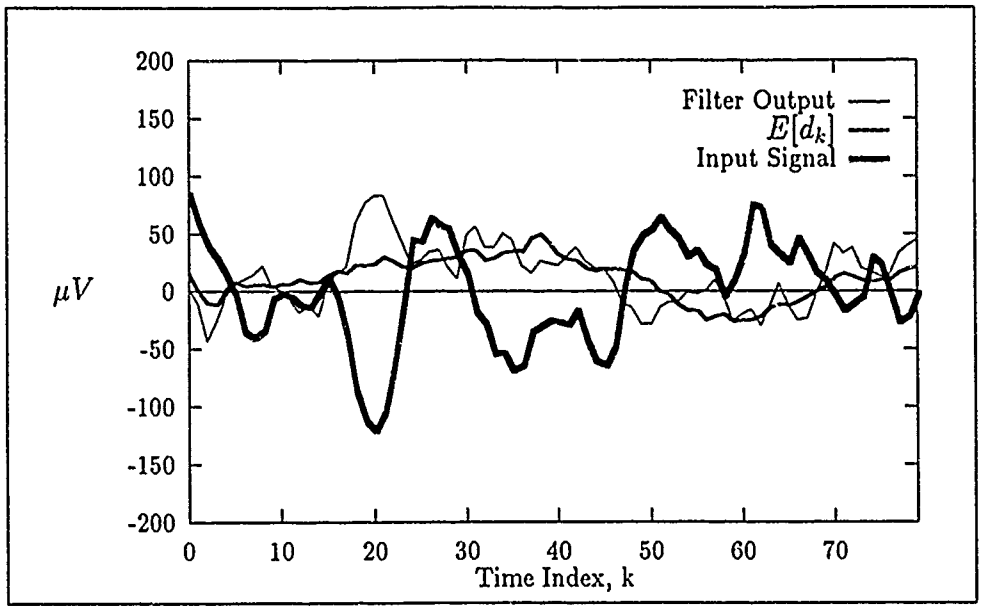

Figure 4.10. Two Sensor $TSAF_{mP_2}$ Results: Post-stimulus output vector $y_{aug,10}$.

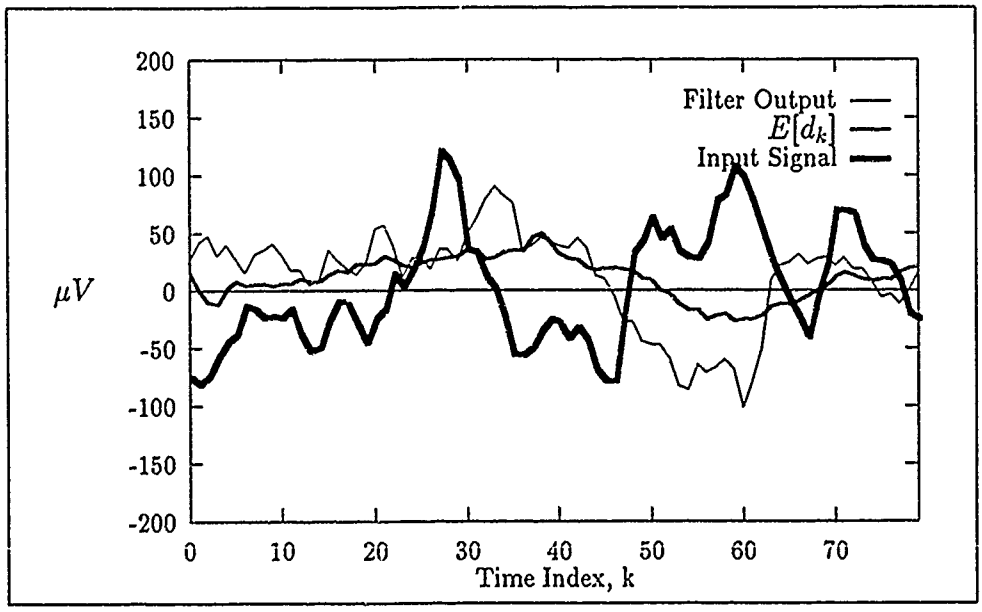

Figure 4.11. Two Sensor $TSAF_{mP_2}$ Results: Post-stimulus output vector $y_{aug,15}$.

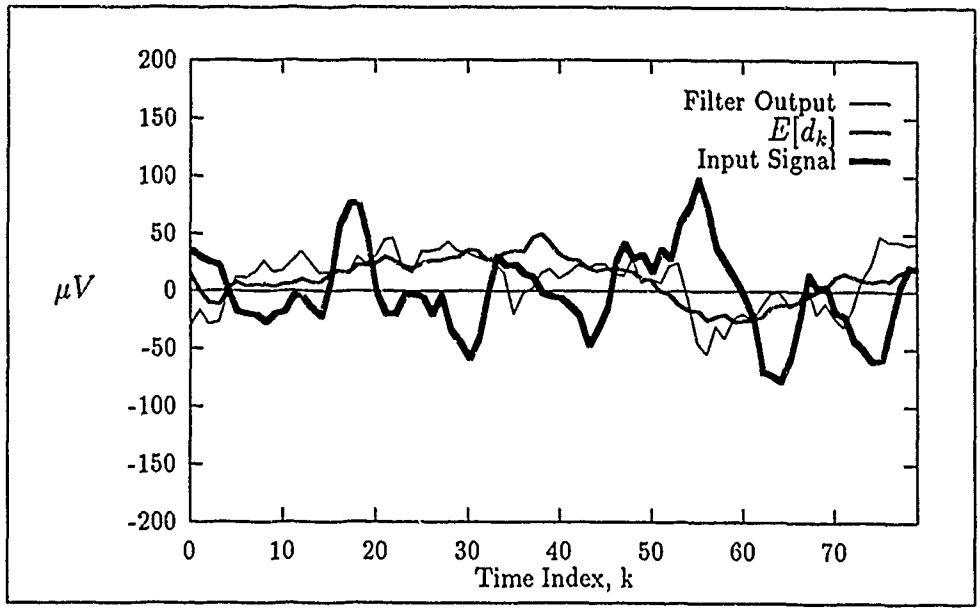

Figure 4.12. Two Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,31}$.

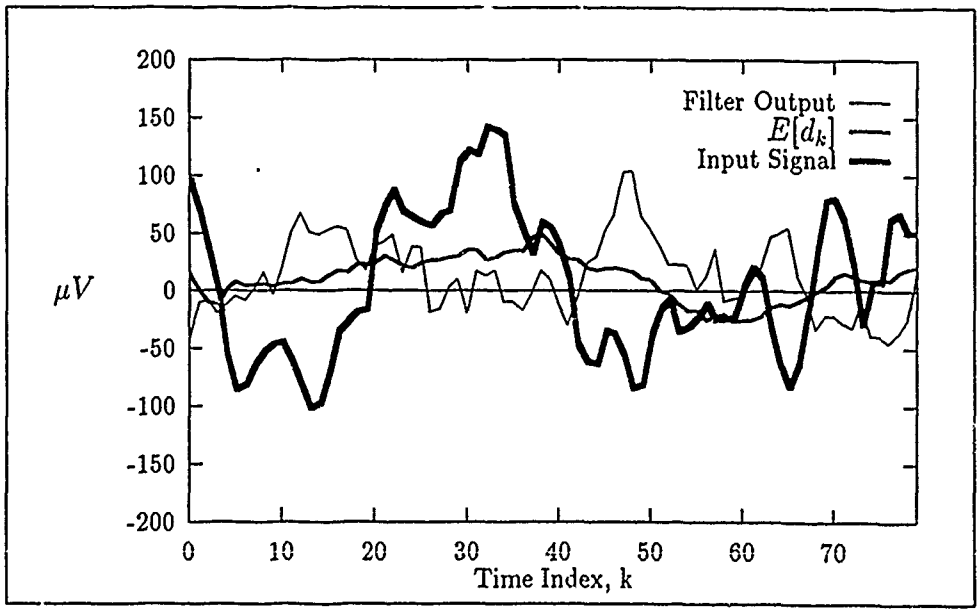

Figure 4.13. Two Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,45}$.

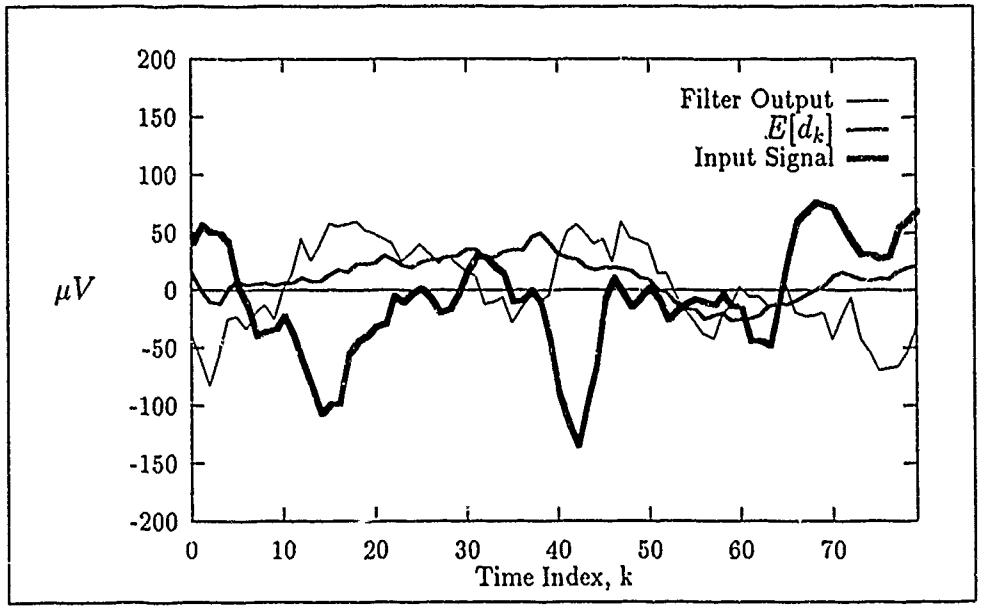

Figure 4.14. Two Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,62}$.

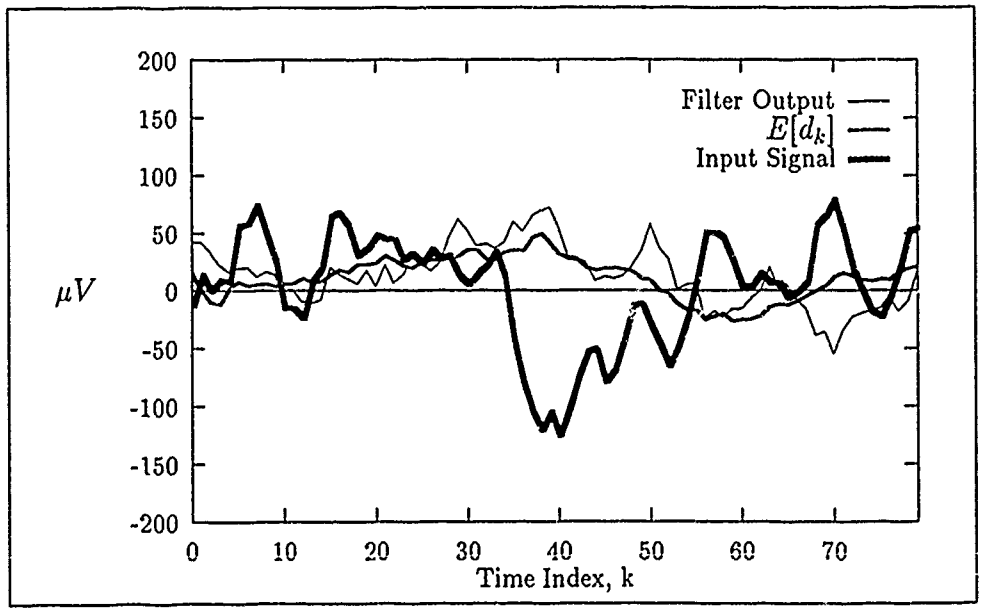

Figure 4.15. Two Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,70}$.

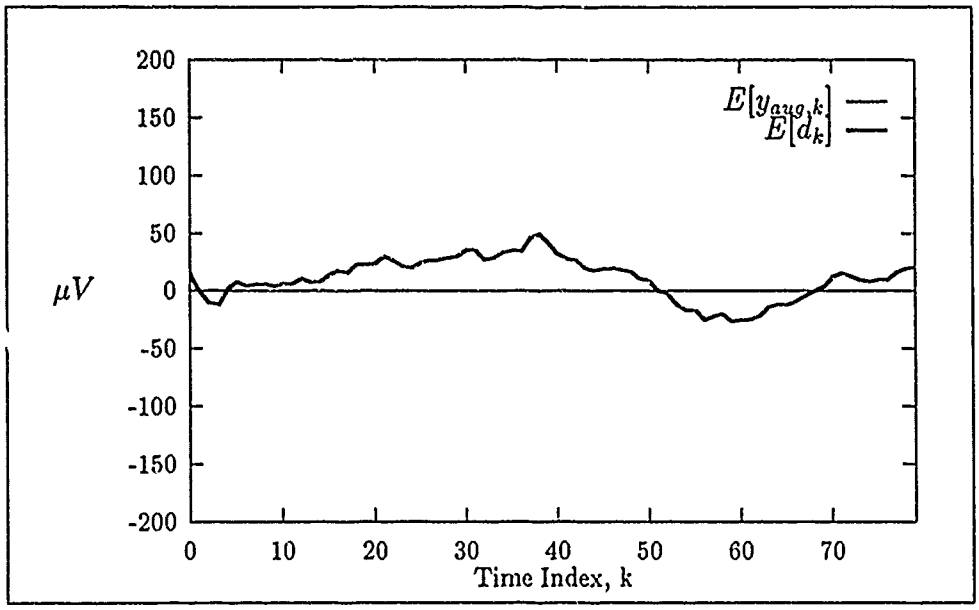

Figure 4.16. Two Sensor $TSAF_{mPa}$ Results: Ensemble Averages of YOUT-EXP2.PRN and MEG.PRN.

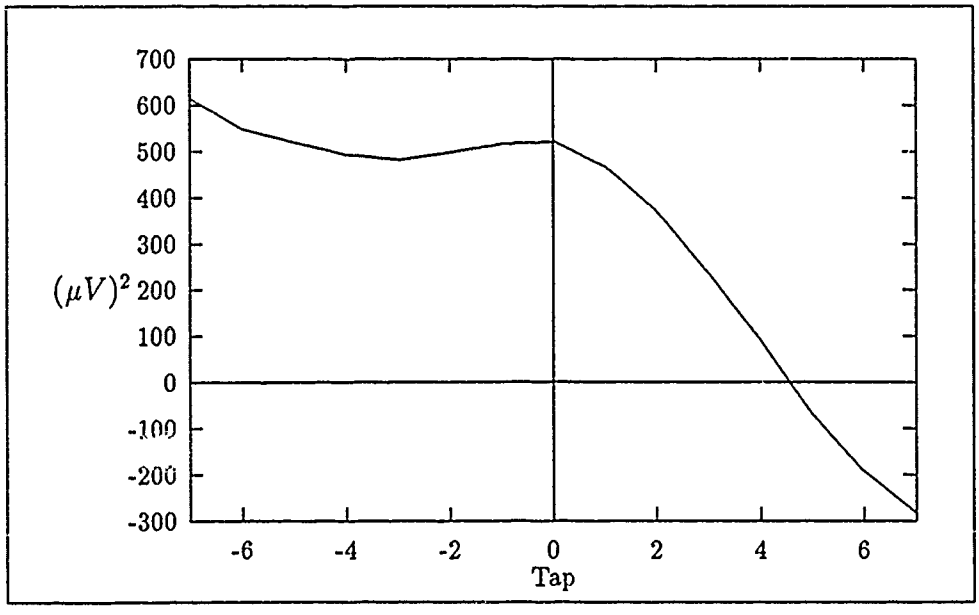

Figure 4.17. Two Sensor $TSAF_{mPa}$ Results: P_n Vector

Finally, before proceeding to the single sensor $TSAF_{mPa}$, six plots follow which compare the $TSAF_{LMS}$ filter output to the $TSAF_{mPa}$ from the previous two experiments. Clearly the two approaches provide different outputs reflecting the contribution of the P_n vector to the estimate of the EF signal in the $TSAF_{mPa}$. The conjecture that the $TSAF_{mPa}$ results are better is only justified based on the simulation results of Chapter 3.

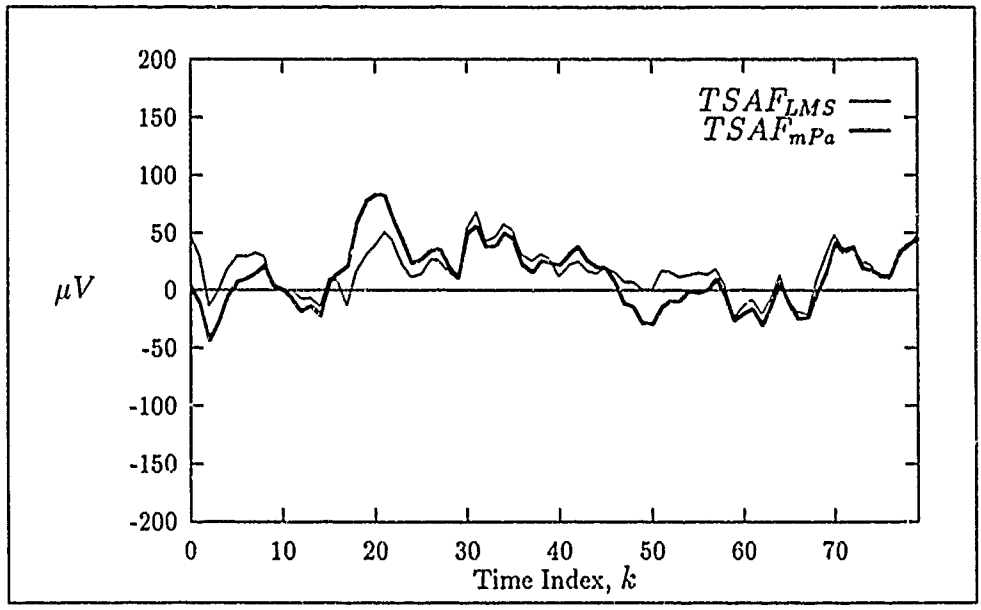

Figure 4.18. Comparison of $y_{aug,10}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments

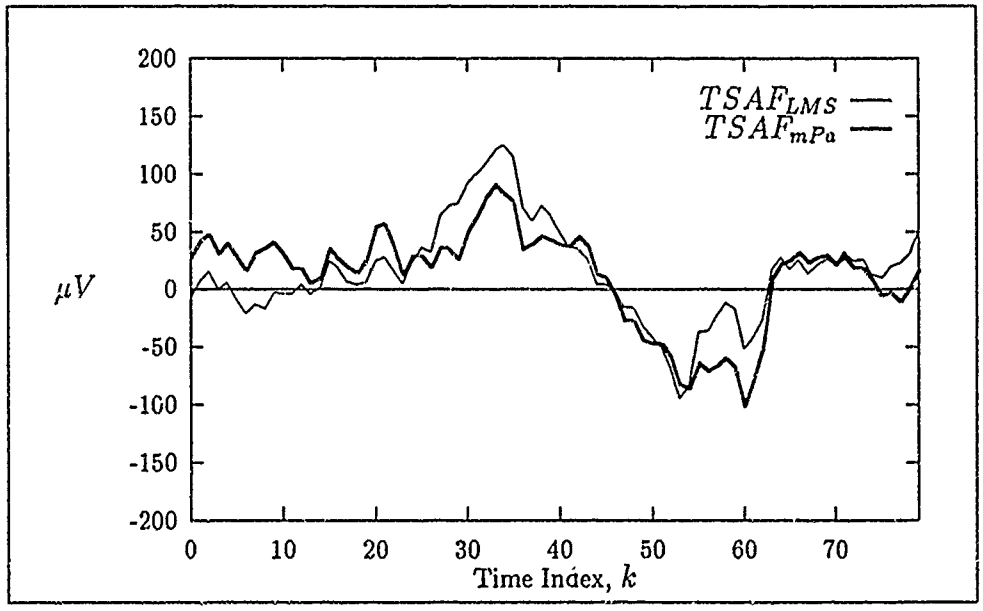

Figure 4.19. Comparison of $y_{aug,15}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments.

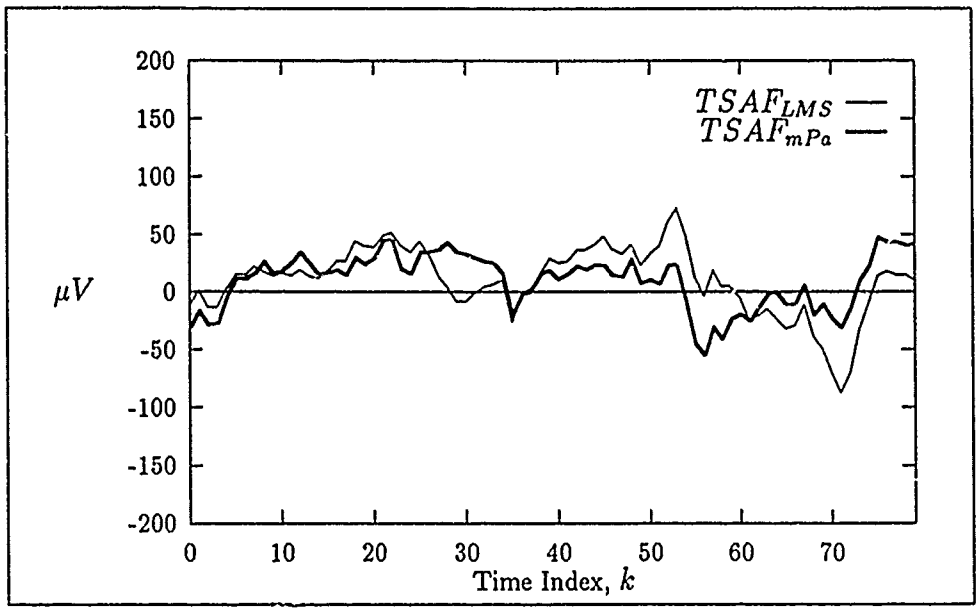

Figure 4.20. Comparison of $y_{aug,31}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments.

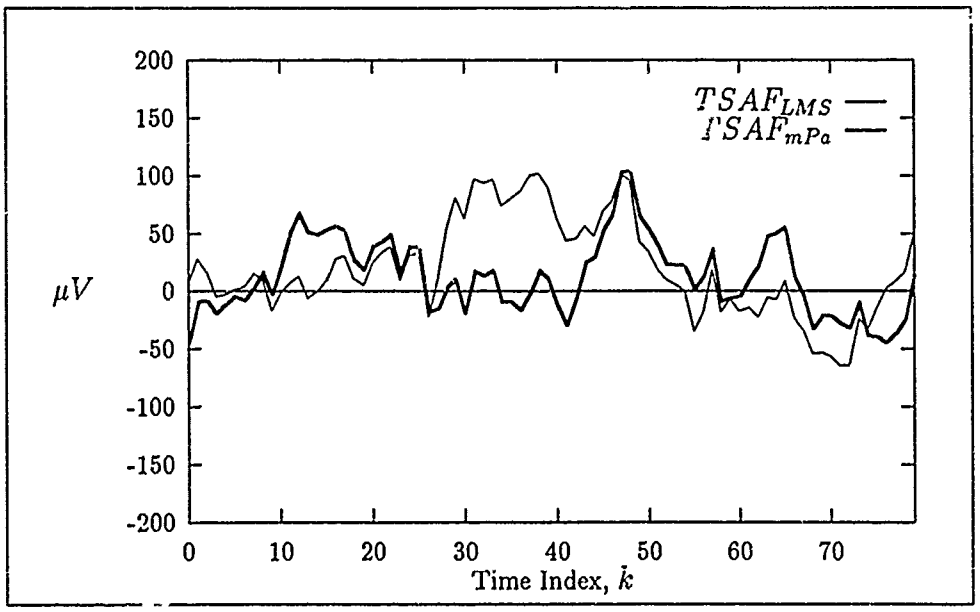

Figure 4.21. Comparison of $y_{aug,45}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments.

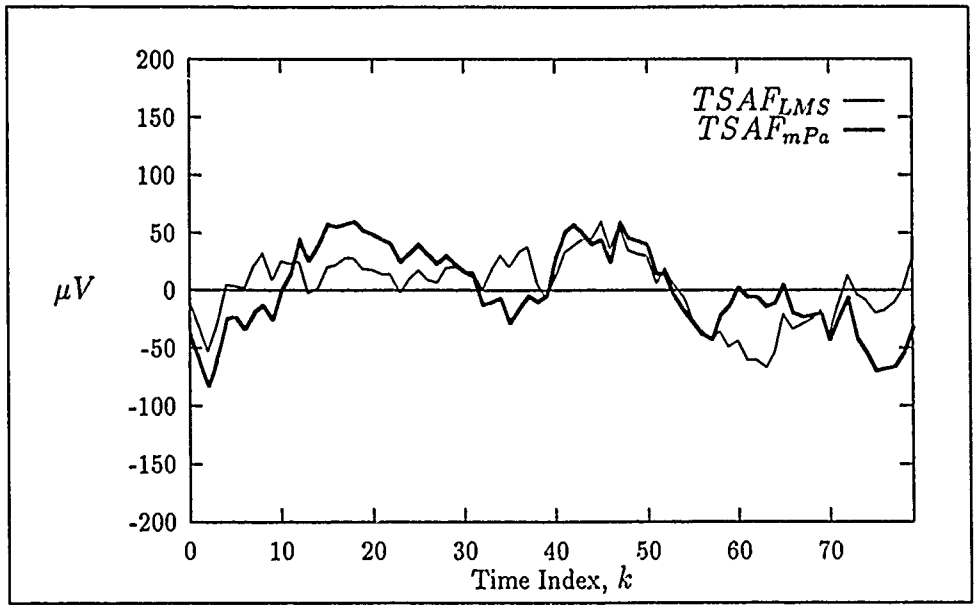

Figure 4.22. Comparison of $y_{aug,62}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments.

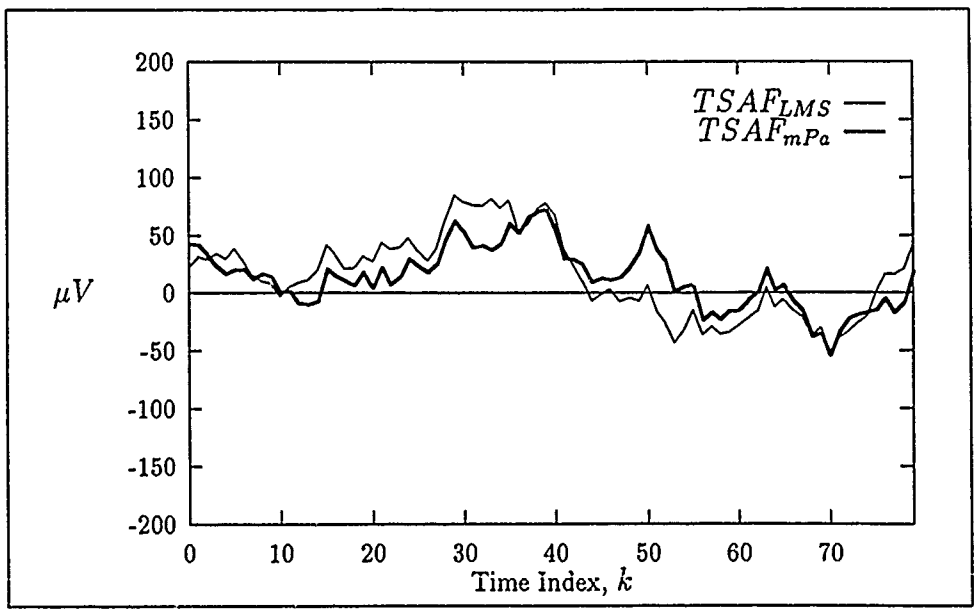

Figure 4.23. Comparison of $y_{aug,70}$: Post-stimulus output vector from two sensor $TSAF_{LMS}$ and $TSAF_{mPa}$ experiments.

4.5 Single Sensor $TSAF_{mPa}$ Estimation of Human EF

This final experiment uses the $TSAF_{mPa}$ with a single sensor with the human EF data for the input and desired signals. Again, this experiment assumes that the noise statistics are time invariant.

4.5.1 Configuration. The filter configuration is shown in Figure 4.24. The P_n vector now estimates the cross correlation statistics of the MEG noise with itself and then removes the estimate in the weight update algorithm. The filter settings are shown in Table 4.3 and the results are presented in the following section.

4.5.2 Results. The results from this experiment was an output file, YOUTEXP3.PRN which contained 80 vectors with 80 discrete points in each and was an estimate of the EF signal component. Six output vectors are shown in Figures 4.25 through 4.30. The ensemble average is plotted to provide a reference along with the input signal used to generate the filter output. As done previously, the ensemble average of the YOUTEXP3.PRN file is compared to the ensemble average of MEG.PRN in Figure 4.31 and the plots were identical. The last piece of data for this experiment is a plot of the P_n vector shown in Figure 4.32 which is very similar to the plot of the autocorrelation of the MEG noise in Figure 3.27. This is expected as the P_n vector represents an estimate of the cross correlation of the MEG noise with itself in the single sensor case.

The plots of the filter output show that the filter was amplifying the input signal. This amplification is undesirable and indicates the filter solution was incorrect. The power of the output signal should have been lower as the filter tries to reduce the effects of the noise contained in the input signal. The next section analyzes the single sensor configuration in terms of the P_n vector.

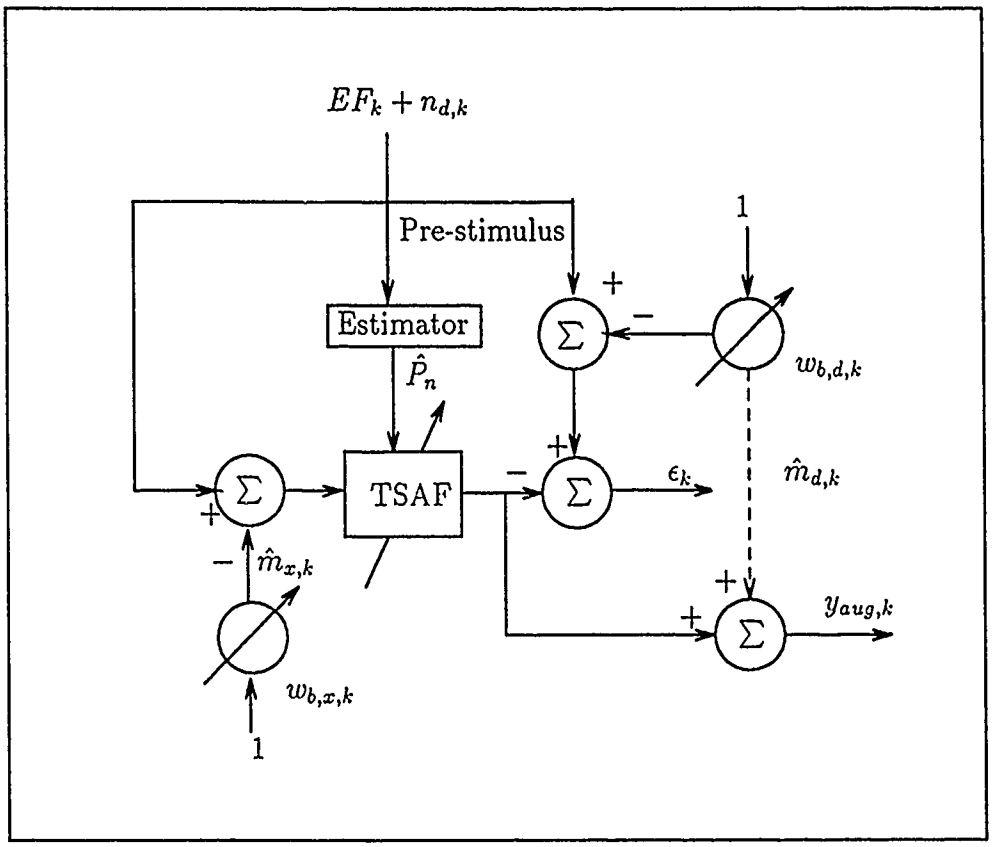

Figure 4.24. Single Sensor $TSAF_{mPa}$ Filter Configuration

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 4.3. Single Sensor $TSAF_{mPa}$ Filter Settings.

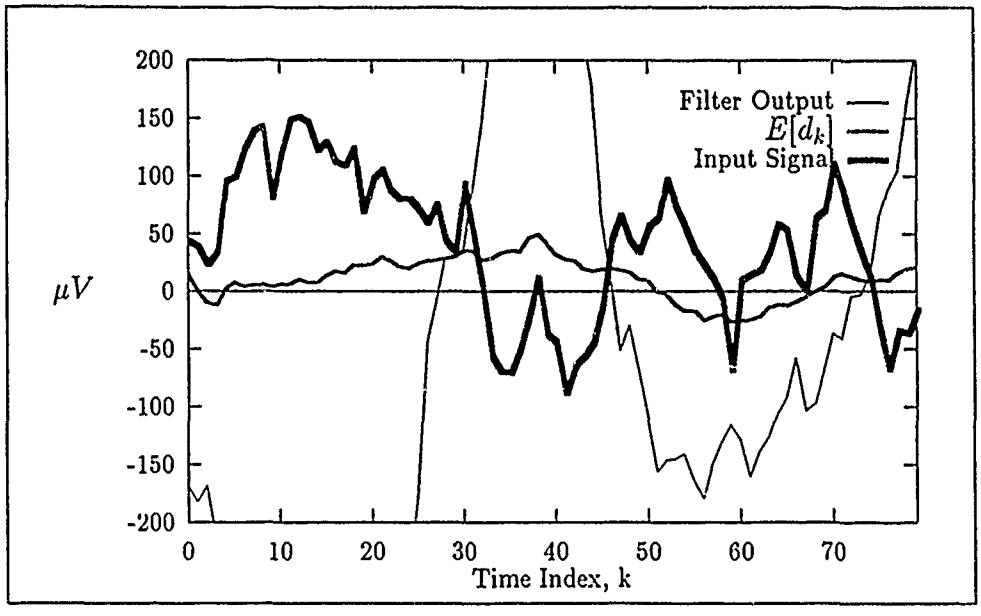

Figure 4.25. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,10}$.

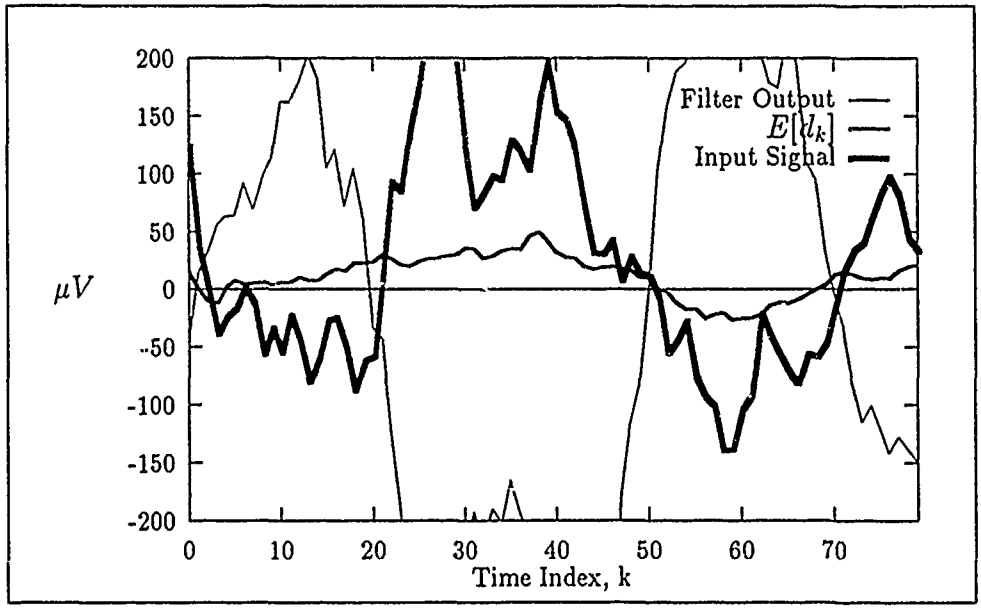

Figure 4.26. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,15}$.

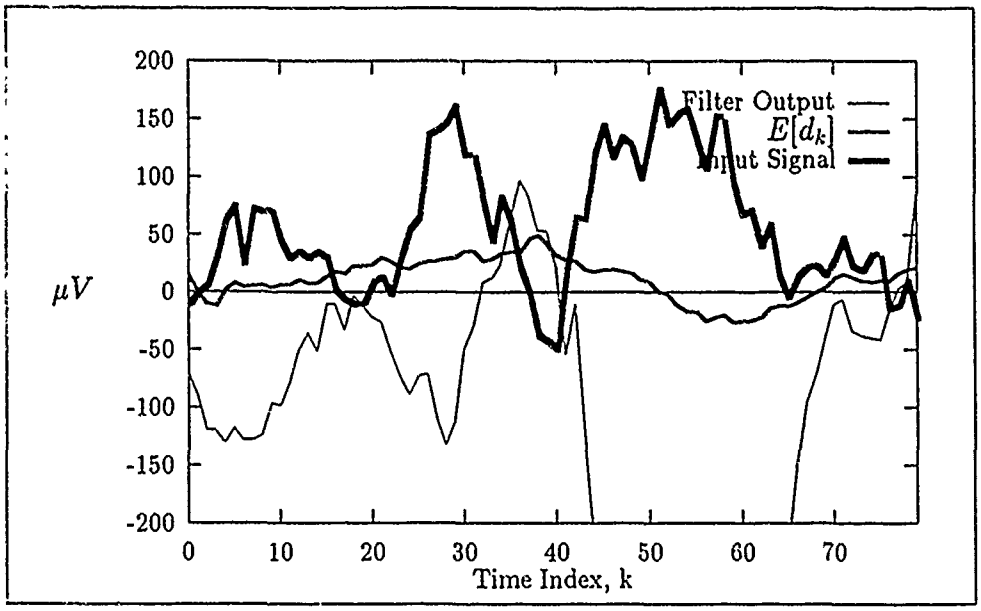

Figure 4.27. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,31}$.

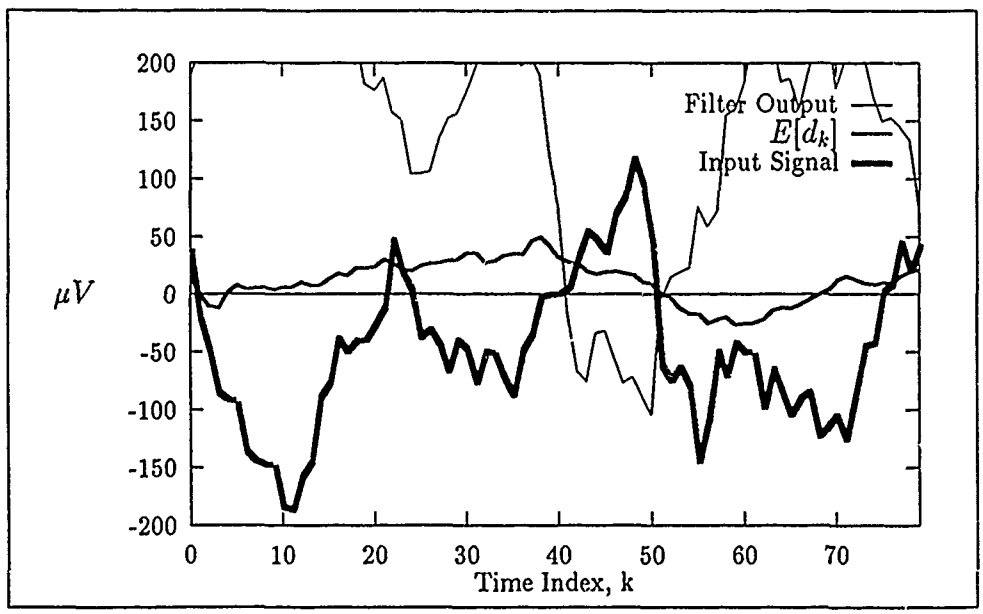

Figure 4.28. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,45}$.

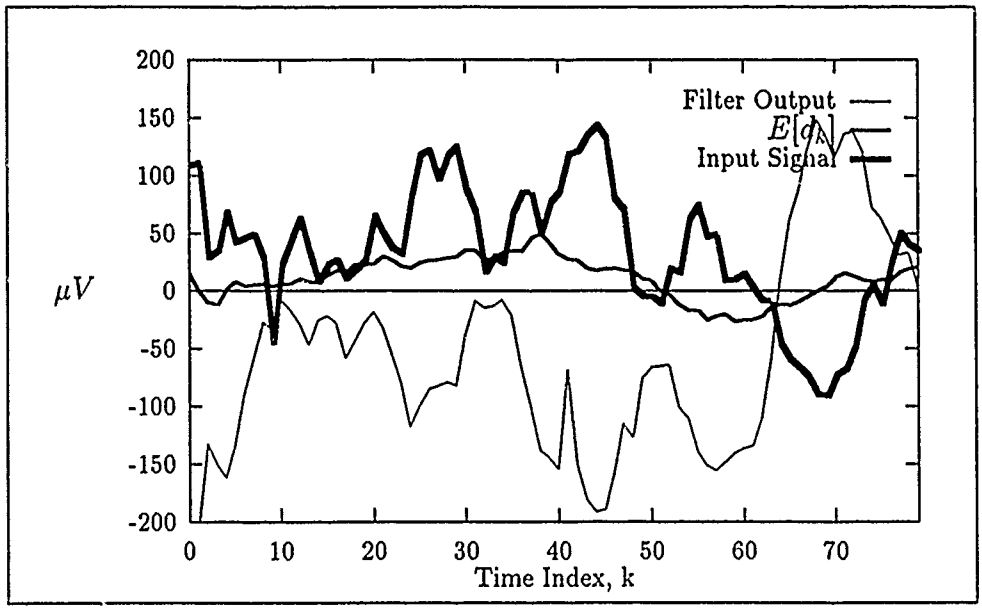

Figure 4.29. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,62}$.

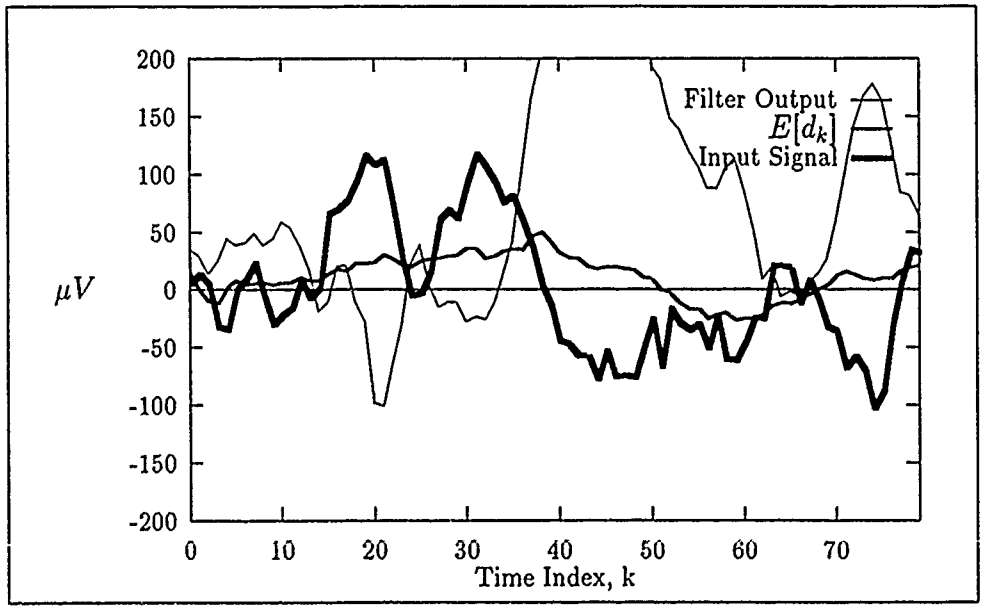

Figure 4.30. Single Sensor $TSAF_{mPa}$ Results: Post-stimulus output vector $y_{aug,70}$.

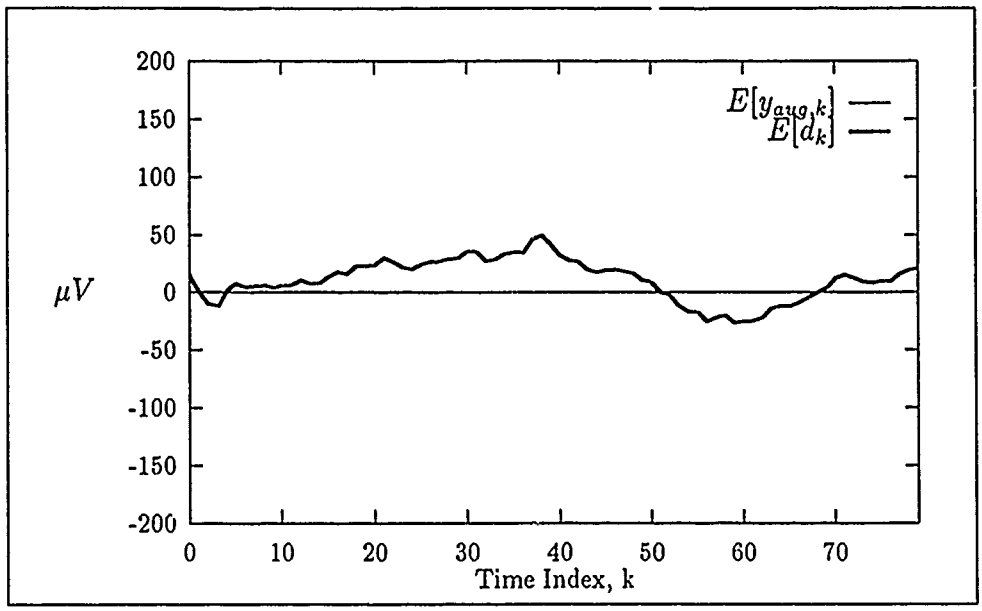

Figure 4.31. Single Sensor $TSAF_{mPa}$ Results: Ensemble Average of YOUT-EXP4.PRN and MEG.PRN

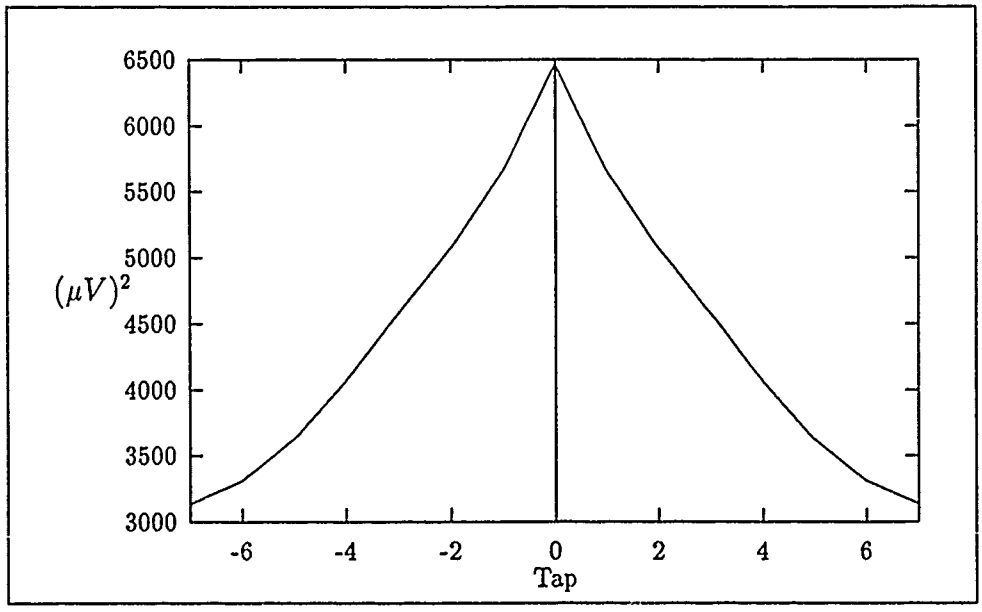

Figure 4.32. Single Sensor $TSAF_{mPa}$ Results: P_n Vector

4.6 Analysis of Single Sensor $TSAF_{mPa}$ Performance

The previous experiment used the single sensor $TSAF_{mPa}$ to estimate human EF signals. The results indicated that the filter was actually amplifying the input signal. This amplification is undesirable as the filter should be reducing the noise power contained in the input signal. This section investigates the performance of the single sensor $TSAF_{mPa}$ in estimating the human EF.

4.6.1 P_n Estimator Revisited. In analyzing the output from the P_n estimator, it was noted that the plot of the P_n vector for the single sensor $TSAF_{mPa}$ experiment was similar, but, not identical to that predicted by the autocorrelation of the human MEG noise performed in Chapter 3. Figure 4.33 compares the P_n vector components to the results from the statistical calculation. Note that the P_n values increase compared to the calculated values as one moves away from the zero tap. The other point to make is that the P_n vector is non-causal with 15 taps and only has values out to ± 7. One might expect the two plots to be identical where they overlap as both are the autocorrelation of the human MEG pre-stimulus noise. The difference is attributed to the "windowing" of the data in the statistical analysis as shown next.

The equation used to calculate the cross correlation statistics is the following:

$$E[d_{j,k}x_{j,k-i}] = \frac{1}{MN}\sum_{0=j}^{M-1}\sum_{0=k}^{N-1} d_{j,k}x_{j,k-i} \tag{4.1}$$

j is the vector pointer, M is the number of vectors contained in the ensemble, k is the time index, N is the number of data points in each vector, and i is the shift index and ranges from $0 \ldots (2N-1)$. The equation form of the P_n estimator is as follows:

$$p_k = \frac{1}{M(N-i)}\sum_{0=j}^{M-1}\sum_{0=k}^{N-1} d_{j,k}x_{j,k-i} \tag{4.2}$$

The $\frac{1}{N}$ term in Equation 4.1 windows the data vectors. This is best seen when the shift index is at its maximum value $i = 2N-1$ and only a single term results from

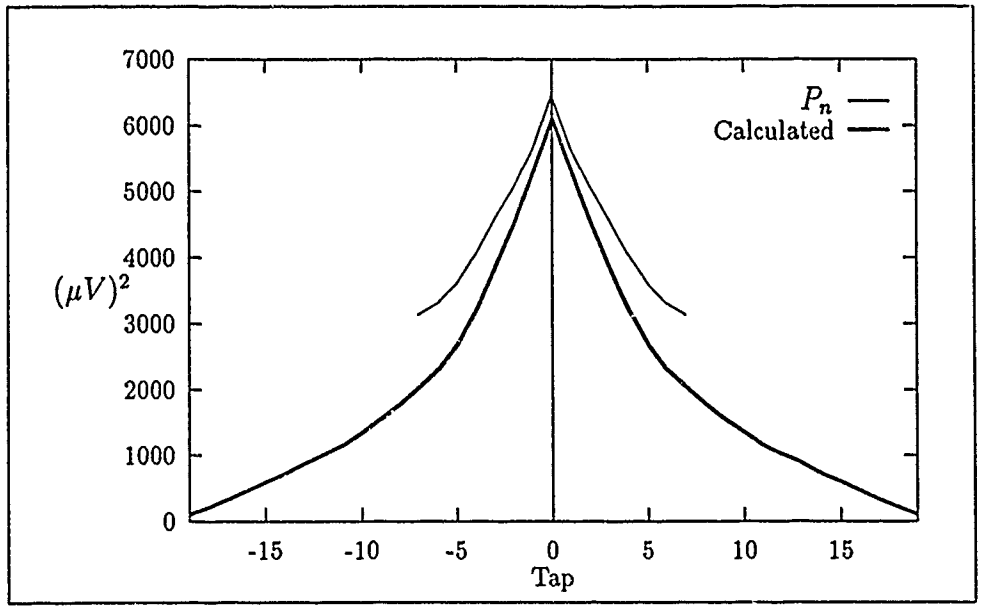

Figure 4.33. Comparison of P_n and Autocorrelation of Human MEG noise

the summation over k as all other values are zero. However, this single term is still divided by N which is the number of data points in the vector.

On the other hand, Equation 4.2 divides the summation over k using the $\frac{1}{N-i}$ term which results in division by the number of terms which overlap. For the same case previously considered where $i = 2N - 1$, the single non-zero term in the summation over k is divided by $N - i$ or 1. Therefore, the P_n estimator equally weights each estimate even though fewer terms are used in the summation. The question then arises as to which equation produces better result in terms of filter performance for the special case investigated in this thesis. The answer was obtained by re-testing the single sensor $TSAF_{mPa}$ with high variance noise added to in the simulated EF data to determine the affects to the filter performance.

4.6.2 $TSAF_{mPa}$ Performance. This section presents two additional test of the single sensor $TSAF_{mPa}$. The test were designed to determine if windowing the data vectors improved the filter performance over using the existing estimator

algorithm. The figure of merit is the average square error using Equation 2.10 from Chapter 3. A new noise file, NOISE2.PRN, was generated as shown in Appendix E. The noise was generated with correlation and high variance as compared to computer generated noise files previously used. The reason for increasing the variance was to more accurately represent the SNR of the human data.

The configuration for both test is shown in Figure 4.34 with the filter settings shown in Table 4.4. These were identical to those used in the estimation of human EF. The MSE results from the test are shown in Table 4.5 and clearly indicate the "window" algorithm had a lower MSE performance than the original estimator. A plot of the resulting P_n vectors is shown in Figure 4.35 and indicates the statistical algorithm matched the calculated autocorrelation of the pre- stimulus noise exactly, where they overlap, while the original algorithm was slightly higher. Output vectors generated from using the different algorithms are plotted, along with the input signal, in Figures 4.36 and 4.37.

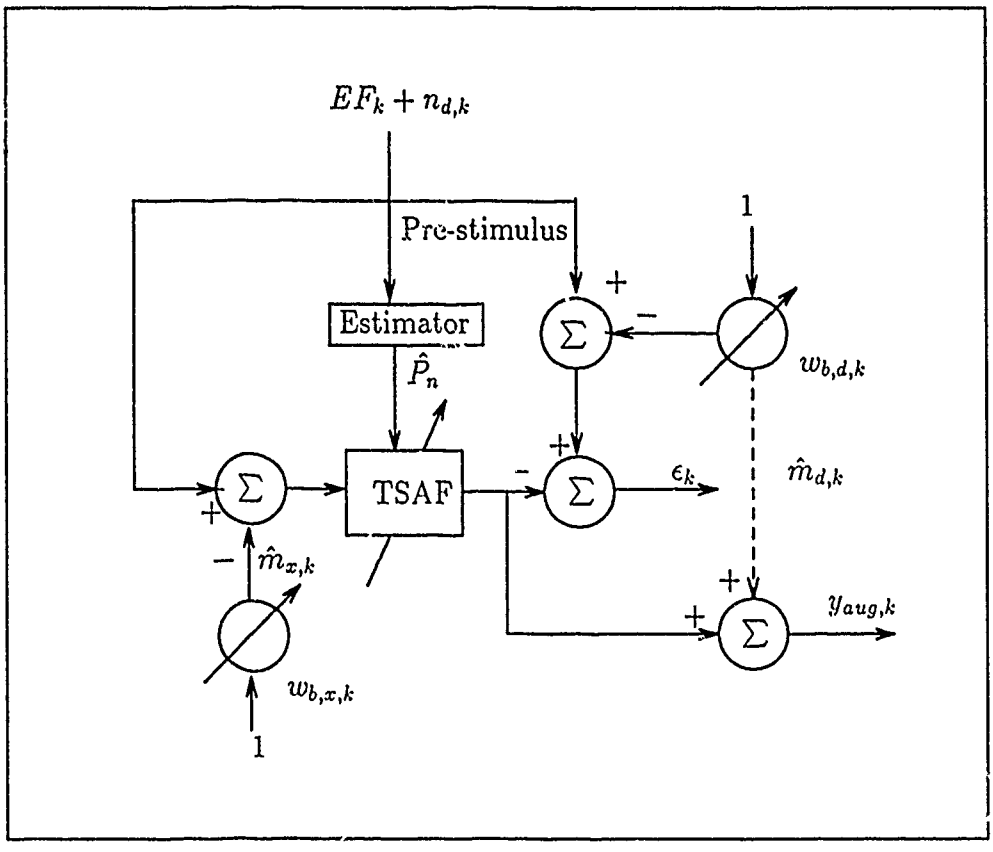

Figure 4.34. P_n Estimator Re-test Filter Configuration. The input is simulated EF with high variance noise, $n_{d,k}$.

Parameter	*Setting*
Number of Runs	10
Number of Taps	15
Misadjustment	0.5
Mode	Non-causal
Algorithm	mPa

Table 4.4. P_n Estimator Re-test Filter Settings

Estimator Algorithm	Average Square Error $(uV)^2$
Estimator Algorithm $(\frac{1}{N-i})$	1145
Statistical Algorithm $(\frac{1}{N})$	779

Table 4.5. P_n Estimator Re-test Results

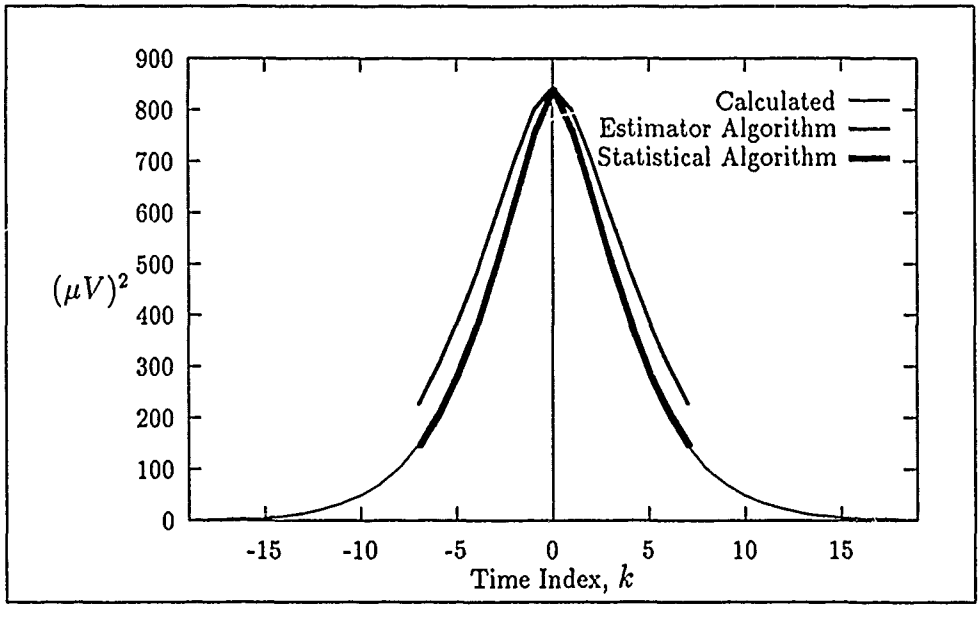

Figure 4.35. P_n Estimator Re-test Results: Comparison of the P_n vectors and calculated autocorrelation of the pre-stimulus noise.

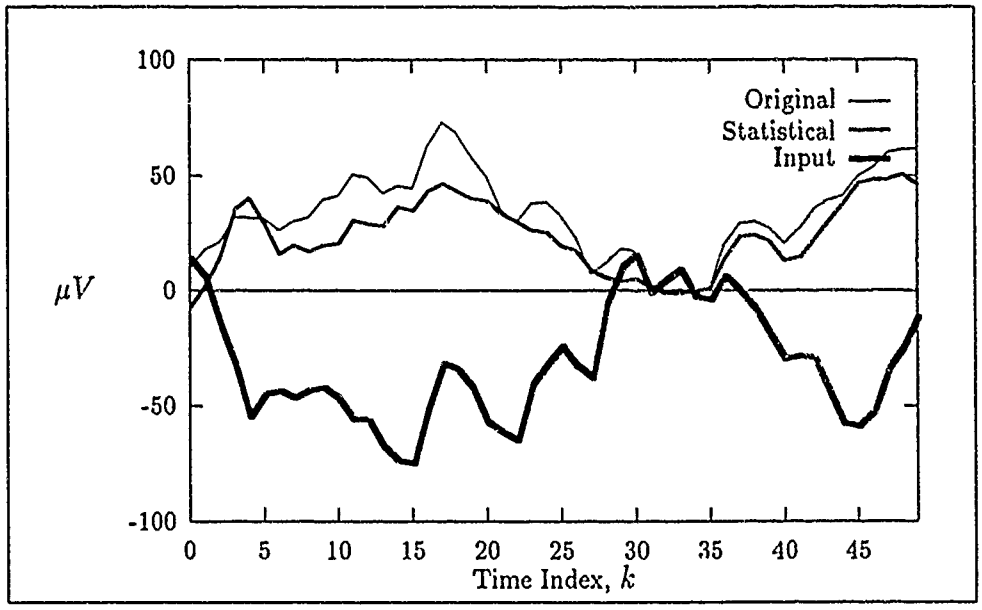

Figure 4.36. P_n Estimator Re-test Results: Output Vector $y_{aug,6}$

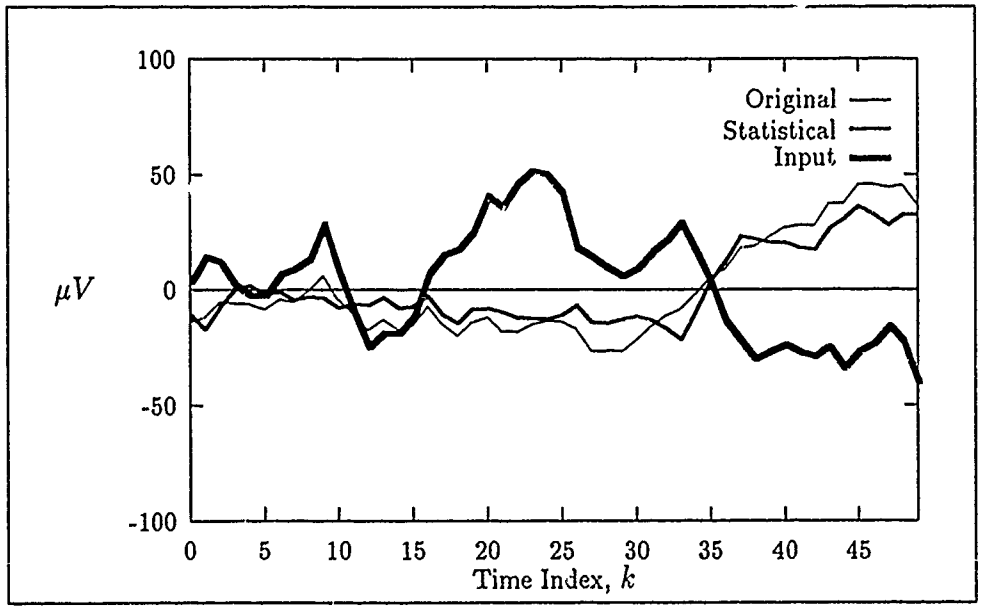

Figure 4.37. P_n Estimator Re-test Results: Output Vector $y_{aug,21}$

4.6.3 Discussion. Based on this limited analysis, windowing improved the single sensor $TSAF_{mPa}$ performance using the simulated EF data with high variance noise. One might attribute this to the outer taps in the estimator having fewer samples from which to estimate the true cross correlation statistics. This results in those taps having a higher error as compared to the taps close to the zero tap which see more samples. In this case, windowing the data appeared to reduce the contribution of the outer taps in the filter update equation and, thus, increased the filter performance. However, repeating the single sensor experiment using the new statistical algorithm with the human EF still resulted in the input signal being amplified.

The single sensor configuration is unique in that the filter relies upon an accurate estimate of the pre-stimulus noise autocorrelation statistics. In addition, the statistics are assumed to be time-invariant. The importance of these assumptions are that $\hat{P}_n = P_n$. Any violation of these assumptions results in $\hat{P}_n = P_n + P_{e,k}$ where $P_{e,k}$ represents the error in the estimate. It can be shown that $W_k^{\#} = W_k^* + W_{e,k}^B$ where the weight error bias, $W_{e,k}^B$, equals $R_k^{-1} P_{e,k}$ (13:89-90). In closing, the following observations were noted during the experiments:

- The magnitude of the P_n vector from the single sensor experiment was extremely high as compared to all other test and experiments performed. In this case, the P_n vector may dominant the filter update equation suggesting that the noise variance exceeds some "threshold" of the filter's ability to see the jitter component.
- The error in the P_n vector is expected to be higher in the outer taps which use fewer pre-stimulus data points. Given the magnitude of the P_n vector for the single sensor experiment, the noise in the outer taps may still be significant even with windowing the pre-stimulus data vector.

- Any time-variance in the noise statistics in the post-stimulus region will corrupt the filter solution as the P_n vector is not time-varying. The ideal estimator would continually update even in the post-stimulus region.

4.7 Chapter Summary

This chapter performed three experiments to estimate the human EF signal from human EP. The three filter configurations used were

1. Two Sensor $TSAF_{LMS}$
2. Two Sensor $TSAF_{mPa}$
3. Single Sensor $TSAF_{mPa}$

The results from these experiments were data files which contained the 80 output vectors from each of the experiments. Unfortunately, there are no human EF signals which are noiseless to compare with the filter output. Therefore, the ensemble average of the desired signal, MEG.PRN, was included in the plots to show the deviation of the output signal about the mean. In addition, the input signal was included to show the reader what the signal looked like before filtering.

Based on the results from Chapter 3, one might reasonably assume that the $TSAF_{mPa}$ would produce a more accurate estimate due to the bias influence of the cross correlation between the MEG and EEG noise. A qualitative assessment of the $TSAF_{mPa}$ performance using the human EF is not possible because the underlying signal is not known.

Finally, the single sensor $TSAF_{mPa}$ experiment using human EF resulted in an output signal which was apparently amplified. This indicated the filter solution was unacceptable and was further analyzed by testing the single sensor $TSAF_{mPa}$ with simulated EF and high variance noise. The results from the test indicated that windowing the pre-stimulus data improved the filter performance but did not resolve the amplification problem. Essentially, this anomaly is left for future research with the cause attributed to three hypothesists:

1. The magnitude of the P_n vector resulting from the autocorrelation of the MEG pre-stimulus noise dominates the filter update equation. This exceeds some "threshold" of the filter's ability to see the jitter component.

2. While windowing improved the filter performance, there may still be considerable error contained in the outer taps of the P_n vector due to the limited number of data points these taps use.

3. The noise in the post-stimulus region may, in fact, be slightly time-varying. This would corrupt the P_n estimate as the statistics are assumed to be time-invariant.

V. Conclusions and Recommendations

5.1 Conclusions

This thesis was a direct application of the Ferrara TSAF using the LMS algorithm and the Williams mPa. The filters were used to estimate human EF signals using human EP. The conclusions from this research effort are the following:

- The input signals to the filter can be modeled as the sum of three uncorrelated components, the jitter, mean, and noise. This was developed in Chapter 2 using the simulated EP data which contained human EEG noise. This result greatly enhanced the understanding of the signal processing performed by the TSAF filters.
- Based on a limited data set, cross correlation was discovered between the human EEG and human EEG noise components. This was based on a statistical analysis of the pre-stimulus noise contained in the human data files obtained from AAMRL. This prompted a series of test which compared the $TSAF_{mPa}$ performance to the $TSAF_{LMS}$ when there is cross correlation between the noise components.
- The $TSAF_{mPa}$ produced a better estimate of the desired signal over the $TSAF_{LMS}$ when the input and desired signals contained correlated noise components. The conclusion was based on using a forward model to create simulated EF from the simulated EP data. In addition, computer generated noise files were used which contained cross correlation.

Other significant observations were made during the testing phase of the filters resulting in modifications to the code are the following:

- The bias weight in the first stage passed adaptation noise into the filter structure which caused some of the filters to diverge from the desired solution. The

filter adaptation was delayed to allow the bias weight to work on the first 20 vectors before the TSAF adaptation started. This reduced the adaptation noise passed to the filter allowing all the TSAF filters to converge to the desired solution.

- The μ_k update algorithm was modified to calculate the gain constant based on an instantaneous estimate of the signal energy present at the filter input. The original algorithm used a leaky integrator stage which captured any noise passed to it from the estimate of the signal energy. This noise was then used in future updates of μ_k which caused the gain constant to decrease in magnitude slowing the convergence of the filter. The instantaneous update appeared to improve the convergence speed to the desired solution.

5.2 Recommendations

There are several recommendations which might make a reasonable thesis topic in themselves or good topics to include in related research. The recommendations are as follows:

1. This thesis analyzed a limited sample of human EEG and MEG data noting there was cross correlation between the noise components. One might further investigate these findings by analyzing the MEG and EEG noise from a much larger sample set and using different subjects and/or different placement of the sensors.
2. This thesis used simulated EF and EP buried in simulated EEG and MEG noise. A follow on thesis should further test the filter using simulated EF and EP buried in *human* EEG and *human* MEG. The purpose being to more closely simulate the signal-to-noise and correlation relationships between the human signals and noise.

3. The performance of the single sensor $TSAF_{mPa}$ could be quantified in terms of MSE as a function of the signal-to-noise ratio. The purpose being to determine if the single sensor $TSAF_{mPa}$ exhibits a threshold affect as the magnitude of the P_n vector components increase. One might also investigate using different windowing techniques in the P_n estimator and study the affects of increasing the number of pre-stimulus data points. For example, generate a 15 tap P_n vector using 30 to 40 pre-stimulus samples instead of the 20 used in this thesis.

4. While signal-to-noise is one performance criteria, another is convergence speed. Therefore, a follow on study should investigate Figures 3.8 and 3.9 to gain additional insight into the effects of the *Counter* performance versus the leaky integrator. Given the filter structure is composed of several components which all interact, additional analysis of the two stage filter might reveal other enhancements to speed the convergence.

5. Along the lines of the previous item, the code for this program could easily be optimized to increase the execution speed. In addition, a user interface would be a nice to have as well. Presently, the code is modified for different configurations and then re-compiled. The program is initialized with variables which could easily be set from within a user interface. Finally, with the speed of PC's increasing, a graphic driver displaying the instantaneous mean-square error would allow monitoring the filter performance during the program execution.

6. Finally, one might investigate using a neural network to improve the filter performance by selecting the optimum filter solution from a set of solutions for each instant in time.

Appendix A. *Definition of Statistics*

A.1 Introduction

The purpose of this appendix is to define terms and describe the statistical calculations referenced through out this thesis. The terms defined are: mean, mean-square, ensemble average, and ensemble mean-square.

A.2 Definitions

Let y_k be a data vector with N data points where k is the time index. The vector represents data points observed from a single experiment or random process over time. The time average or mean of Y is defined as

$$\hat{y} = \frac{1}{N}\sum_{k=1}^{N} y_k \tag{A.1}$$

It is important to note that if N is finite, Equation A.1 is only an estimate of the mean. However, for the purpose of this thesis, the number of data points is considered to be large enough such that:

$$\bar{y} \simeq \hat{y} = \frac{1}{N}\sum_{k=1}^{N} y_k \tag{A.2}$$

Continuing on and using the same data vector, the mean-square of y is defined as

$$\bar{y^2} = \frac{1}{N}\sum_{k=1}^{N} y_k^2 \tag{A.3}$$

Both of these are commonly used in calculating the statistics of time sequenced data (1:3-5). The point to make is that the statistics were calculated over time or "along" the process. We are now ready to define the ensemble average which is calculated "across" the process.

The experiment is now repeated M times still making N observations during each experiment. The observations are collected and stored in the data array $y_{j,k}$ which now has M rows and N columns. k is still the time (column) index and j is the vector (row) index. This data array then represents the ensemble of the experiment (7:114-115). When dealing with ensembles, the statistics of interest are the ensemble average and the ensemble mean-square. These are calculated "across" the process at fixed points in time. Therefor, the ensemble average is calculated by columns in the data array or the ensemble average is

$$E[y_k] = \frac{1}{M}\sum_{j=1}^{M} y_{j,k} \tag{A.4}$$

where $k = 1, 2, ...N$ and indicates that the ensemble average is time sequenced and with N values as a result of the calculation. Note that the averaging is done by columns. In this thesis, the resulting values are represented in vector notation. Therefor, the N results from Equation A.4 can be written as

$$E[Y] = [E[y_1]\ E[y_2]...E[y_N]]^T \tag{A.5}$$

where Y is a vector with N data points. The ensemble mean-square is then

$$E[y_k^2] = \frac{1}{M}\sum_{j=1}^{M} y_{j,k}^2 \tag{A.6}$$

The notation in this thesis uses $E[y_k]$ as the ensemble average and $E[y_k^2]$ as the ensemble mean square or variance of the k^{th} column in the data ensemble.

Appendix B. *Bias Weight Solution*

B.1 Introduction

This Appendix presents the derivation of the equation for the TSAF optimum weight solution with the bias weight. This derivation relies upon the following identity:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BV^{-1}CA^{-1} & -A^{-1}BV^{-1} \\ -V^{-1}CA^{-1} & V^{-1} \end{bmatrix} \tag{B.1}$$

where $V = (D - CA^{-1}B)$ and the determinants of A and V must be non-zero. The proof of this identity can be found in (13:121-123). Only the result is used here.

B.2 Optimum Weight Vector Solution

The derivation begins with the augmented optimum weight vector in Equation 2.27 which is rewritten below:

$$W^*_{aug,k} = \begin{bmatrix} 1 & M_k^T \\ M_k & R_k \end{bmatrix}^{-1} \begin{bmatrix} m_{d,k} \\ P_k \end{bmatrix} \tag{B.2}$$

Applying the identity in Equation B.1 to Equation B.2, the optimum weight vector solution is simplified as follows:

$$W^*_{aug,k} = \begin{bmatrix} (1^{-1} + 1^{-1}M_k^T F_k^{-1} M_k 1^{-1}) & (-1^{-1}M_k^T F_k^{-1}) \\ -F^{-1}M_k 1^{-1} & F_k^{-1} \end{bmatrix} \begin{bmatrix} m_{d,k} \\ P_k \end{bmatrix} \tag{B.3}$$

where

$$A = 1 \tag{B.4}$$

$$B = M_k^T \tag{B.5}$$

$$C = M_k \tag{B.6}$$

$$D = R_k \tag{B.7}$$

F_k is the autocorrelation matrix of the input signal with the mean removed. One can derive F_k from the definition of V which results in

$$\begin{aligned} V &= D - CA^{-1}B \\ &= R_k - M_k 1^{-1} M_k^T \\ &= R_k - M_k M_k^T \\ &= F_k \end{aligned} \tag{B.8}$$

Performing the vector multiplication and grouping terms:

$$\begin{aligned} W^*_{aug,k} &= \begin{bmatrix} m_{d,k} + m_{d,k} M_k^T F_k^{-1} M_k - M_k^T F k^{-1} P_k \\ -m_{d,k} F_k^{-1} M_k + F_k^{-1} P_k \end{bmatrix} \\ &= \begin{bmatrix} m_{d,k} + M_k^T F_k^{-1} (m_{d,k} M_k - P_k) \\ F_k^{-1} (-m_{d,k} M_k + P_k) \end{bmatrix} \end{aligned} \tag{B.9}$$

In order to proceed, the equation for the cross correlation vector must be expanded and written in terms of the individual signal components. This is presented below:

$$\begin{aligned} P_k &= E[d_k X_k] \\ &= E[d_k (M_k + Q_k + N_k)] \\ &= E[d_k M_k] + E[d_k Q_k] + E[d_k N_k] \\ &= E[d_k] E[M_k] + E[d_k Q_k] + E[d_k] E[N_k] \\ &= m_{d,k} M_k + E[d_k Q_k] \end{aligned} \tag{B.10}$$

The last line is obtained by recalling that the noise component of the input signal is zero mean and uncorrelated with the desired signal. In addition, the mean of

the input signal is uncorrelated with the desired signal and the expectation of the product equals the product of the expectations. Finally, Equation B.10 is further reduced by writing d_k in terms of its components,

$$\begin{aligned} P_k &= m_{d,k}M_k + E[d_kQ_k] \\ &= m_{d,k}M_k + E[(m_{d,k} + q_{d,k} + n_{d,k})Q_k)] \\ &= m_{d,k}M_k + E[m_{d,k}Q_k] + E[q_{d,k}Q_k] + E[n_{d,k}Q_k] \\ &= m_{d,k}M_k + E[m_{d,k}]E[Q_k] + E[q_{d,k}Q_k] + E[n_{d,k}]E[Q_k] \\ &= m_{d,k}M_k + E[q_{d,k}Q_k] \\ &= m_{d,k}M_k + Q_{d,k} \end{aligned} \tag{B.11}$$

where $Q_{d,k} = E[q_{d,k}Q_k]$ and is the expected value of the correlated jitter or random components of the input signal and the desired signal.

Now substituting this result into Equation B.9 for the quantities in the parenthesis, the final form of the optimum weight vector solution is obtained:

$$W_k^* = \begin{bmatrix} w_{b,k} \\ W_k \end{bmatrix} = \begin{bmatrix} m_{d,k} - M_k^T F_k^{-1} Q_{d,k} \\ F_k^{-1} Q_{d,k} \end{bmatrix} \tag{B.12}$$

This is the solution presented in Chapter 2.

Appendix C. *Optimum Weight Vector Solution.*

C.1 Introduction.

This appendix derives the filter solutions for the case when the input and desired signal are the same (i.e. $x_k = d_k$). The solutions are developed using a three tap filter for three different cases: causal filter without noise in the filter input, non-causal filter without noise in the filter input, and non-causal with noise added to the filter input. In general, a causal filter processes only current signal values and/or past. Often the terms realizable and causal go hand in hand (4:38). A non-causal filter is a filter which not only processes current and/or past signal values, but, future values as well (5:41). For this thesis, the entire event has already occurred and the sample points are stored in a data vector, therefore, future values are the next sample point(s) in the vector relative to where "now" is. For the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters, "now" is the zero or center tap of the filter, w_0.

C.1.1 Causal Filter Solution. Given a three tap causal filter, one uses the definition of the autocorrelation matrix and cross correlation vector establish the following optimum weight solution:

$$\begin{aligned} W^* &= R^{-1}P \\ &= \begin{bmatrix} E[x_0x_0] & E[x_0x_{-1}] & E[x_0x_{-2}] \\ E[x_{-1}x_0] & E[x_{-1}x_{-1}] & E[x_{-1}x_{-2}] \\ E[x_{-2}x_0] & E[x_{-2}x_{-1}] & E[x_{-2}x_{-2}] \end{bmatrix}^{-1} \begin{bmatrix} E[d_0x_0] \\ E[d_0x_{-1}] \\ E[d_0x_{-2}] \end{bmatrix} \end{aligned} \tag{C.1}$$

To simplify the development of the solution, the following matrix and vector notations are presented:

$$W^* = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}^{-1} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{C.2}$$

where

$$a = E[x_0 x_0] \tag{C.3}$$

$$b = E[x_0 x_{-1}] \tag{C.4}$$

$$c = E[x_0 x_{-2}] \tag{C.5}$$

$$d = E[x_{-1} x_{-1}] \tag{C.6}$$

$$e = E[x_{-1} x_{-2}] \tag{C.7}$$

$$f = E[x_{-2} x_{-2}] \tag{C.8}$$

An important note is that the order of the product within the expected value operator is not important or $E[x_0 x_{-1}] = E[x_{-1} x_0]$. In addition, the input and desired signal are the same and $d_0 = x_0$ which results in the cross correlation vector containing the same values as the top row of the R matrix. The next step is to invert the R matrix and perform the vector product which produces the following:

$$W^* = \frac{1}{Det\ R} \begin{bmatrix} (df - e^2) & (ce - bf) & (be - cd) \\ (ce - bf) & (af - c^2) & (bc - ae) \\ (be - cd) & (bc - ae) & (ad - b^2) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$= \frac{1}{Det\ R} \begin{bmatrix} adf - ae^2 + bce - b^2 f + bce - c^2 d \\ ace - abf + abf - bc^2 + bc^2 - ace \\ abe - acd + b^2 c - abe + acd - b^2 c \end{bmatrix}$$

$$= \frac{1}{Det\ R}\begin{bmatrix} Det\ R \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{C.9}$$

where $Det\ R = adf - ae^2 + bce - b^2f + bce - c^2d$. The inversion assumes the determinant of R is non-zero. This is not a surprising result given the input and desired signal are the same because the best the filter can do, in this case, is model a straight wire and pass the input signal without any alterations. This is the result for a causal filter. The derivation for a non-casual filter now follows.

C.1.2 Non-Causal Filter Solution. The non-causal filter has weights which see past and future values. For this thesis, the non-causal filter is always symmetric about the center weight, w_{0k}. This development will again use a three tap filter which means that the filter uses only one future input. The input vector is then $X = [x_1\ x_0\ x_{-1}]^T$ and the optimum weight solution is written slightly different from Equation B.1 or

$$\begin{aligned} W^* &= R^{-1}P \\ &= \begin{bmatrix} E[x_1x_1] & E[x_1x_0] & E[x_1x_{-1}] \\ E[x_0x_1] & E[x_0x_0] & E[x_0x_{-1}] \\ E[x_{-1}x_1] & E[x_{-1}x_0] & E[x_{-1}x_{-1}] \end{bmatrix}^{-1} \begin{bmatrix} E[d_0x_1] \\ E[d_0x_0] \\ E[d_0x_{-1}] \end{bmatrix} \end{aligned} \tag{C.10}$$

Again, $d_0 = x_0$ and one to one substitutions are made to simplify the expansion of the matrix inverse. Equation C.10 then becomes the following:

$$W^* = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}^{-1} \begin{bmatrix} b \\ d \\ e \end{bmatrix} \tag{C.11}$$

where

$$a = E[x_1 x_1] \tag{C.12}$$

$$b = E[x_1 x_0] \tag{C.13}$$

$$c = E[x_1 x_{-1}] \tag{C.14}$$

$$d = E[x_0 x_0] \tag{C.15}$$

$$e = E[x_0 x_{-1}] \tag{C.16}$$

$$f = E[x_{-1} x_{-1}] \tag{C.17}$$

The important difference to note is the change in the P vector. To ensure the reader sees the time shift from using a non-causal filter, the substitutions used with the P vector are presented with the input signal equal to the desired signal:

$$P = \begin{bmatrix} E[d_0 x_1] \\ E[d_0 x_0] \\ E[d_0 x_{-1}] \end{bmatrix} = \begin{bmatrix} E[x_0 x_1] \\ E[x_0 x_0] \\ E[x_0 x_{-1}] \end{bmatrix} = \begin{bmatrix} b \\ d \\ e \end{bmatrix} \tag{C.18}$$

The non causal filter has shifted the input signal, however, the desired signal is not shifted and the cross correlation vector now contains the values in the second row of the R matrix. Inverting the matrix and performing some algebra, the three tap

non-causal solution is the following:

$$
\begin{aligned}
W^* &= \frac{1}{Det\ R}\begin{bmatrix} (df-e^2) & (ce-bf) & (be-cd) \\ (ce-bf) & (af-c^2) & (bc-ae) \\ (be-cd) & (bc-ae) & (ad-b^2) \end{bmatrix}\begin{bmatrix} b \\ d \\ e \end{bmatrix} \\
&= \frac{1}{Det\ R}\begin{bmatrix} bdf-be^2+cde-bdf+be^2-cde \\ bce-b^2f+adf-c^2d+bce-e^2a \\ b^2e-bcd+bcd-ade+ade-b^2e \end{bmatrix} \\
&= \frac{1}{Det\ R}\begin{bmatrix} 0 \\ Det\ R \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}
\end{aligned}
\tag{C.19}
$$

where $DET\ R = bce - b^2f + adf - c^2d + bce - e^2a$ and is assumed to be non-zero. This shows again that when the input and desired signal are the same, the filter converges to a straight wire where $w_0 = 1$ and all other taps are zero.

C.1.3 Non-Causal Filter Solution With Noise. This section derives the filter solution for the special case where noise is added to the input signal, however, the desired signal is noiseless or

$$x_k = q_k + n_{x,k} \tag{C.20}$$

and

$$d_k = q_k \tag{C.21}$$

The mean components are absent because we have assumed that the bias weights have perfectly estimated the mean components of the input and desired signals and removed them. The next step uses similar substitutions as those already presented

to write the optimum weight vector solution in terms of the autocorrelation matrix and the cross correlation vector or

$$W^* = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}^{-1} \begin{bmatrix} E[d_0 x_1] \\ E[d_0 x_0] \\ E[d_0 x_{-1}] \end{bmatrix} \tag{C.22}$$

where

$$a = E[x_1 x_1] = E[q_1^2 + n_1^2] \tag{C.23}$$

$$b = E[x_1 x_0] = E[q_1 q_0 + n_1 n_0] \tag{C.24}$$

$$c = E[x_1 x_{-1}] = E[q_1 q_{-1} + n_1 n_{-1}] \tag{C.25}$$

$$d = E[x_0 x_0] = E[q_0^2 + n_0^2] \tag{C.26}$$

$$e = E[x_0 x_{-1}] = E[q_0 q_{-1} + n_0 n_{-1}] \tag{C.27}$$

$$f = E[x_{-1} x_{-1}] = E[q_{-1}^2 + n_{-1}^2] \tag{C.28}$$

The cross terms are zero because the signal components are assumed to be uncorrelated and zero mean as was shown in Chapter 2. Therefor, the expected value of the cross terms in the product is zero. The P vector is now written in terms of Equations C.24, C.26, and C.27 or

$$P = \begin{bmatrix} E[d_0 x_1] \\ E[d_0 x_0] \\ E[d_0 x_{-1}] \end{bmatrix}$$

$$= \begin{bmatrix} E[q_0(q_1 + n_1)] \\ E[q_0(q_0 + n_0)] \\ E[q_0(q_{-1} + n_{-1})] \end{bmatrix}$$

$$
\begin{aligned}
&= \begin{bmatrix} b - E[n_{x,0}n_{x,1}] \\ d - E[n_{x,0}^2] \\ e - E[n_{x,0}n_{x,-1}] \end{bmatrix} \\
&= \begin{bmatrix} b - n_b \\ d - n_d \\ e - n_e \end{bmatrix} \qquad \text{(C.29)}
\end{aligned}
$$

The final form of the solution is obtained by performing the inverse operation on the autocorrelation matrix and breaking the P vector into two separate vectors or

$$
\begin{aligned}
W^* &= \frac{1}{Det\ R} \begin{bmatrix} (df - e^2) & (ce - bf) & (be - cd) \\ (ce - bf) & (af - c^2) & (bc - ae) \\ (be - cd) & (bc - ae) & (ad - b^2) \end{bmatrix} \begin{bmatrix} b - n_b \\ d - n_d \\ e - n_e \end{bmatrix} \\
&= \frac{1}{Det\ R} \begin{bmatrix} (df - e^2) & (ce - bf) & (be - cd) \\ (ce - bf) & (af - c^2) & (bc - ae) \\ (be - cd) & (bc - ae) & (ad - b^2) \end{bmatrix} \left(\begin{bmatrix} b \\ d \\ e \end{bmatrix} - \begin{bmatrix} n_b \\ n_d \\ n_e \end{bmatrix} \right) \qquad \text{(C.30)}
\end{aligned}
$$

Performing the vector multiplication and using the results from Equation C.19, the previous expression is simplified to the following:

$$
W^* = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - \frac{1}{Det\ R} \begin{bmatrix} (df - e^2) & (ce - bf) & (be - cd) \\ (ce - bf) & (af - c^2) & (bc - ae) \\ (be - cd) & (bc - ae) & (ad - b^2) \end{bmatrix} \begin{bmatrix} n_b \\ n_d \\ n_e \end{bmatrix} \qquad \text{(C.31)}
$$

Equation C.31 shows that the resulting weight vector will try and reduce the contribution of the noise in the filtered output while enhancing the signal component.

Appendix D. *Derivation of the mPa*

D.1 Introduction

This appendix develops the Williams mPa from the Widrow LMS algorithm. The development follows that used by Williams and presents two derivations. The first derivation shows the effects of the signal noise component in the weight solution when a single sensor is used. The second derivation is the mPa (13).

D.2 The mPa

The derivation starts by expanding the error term,ϵ_k, in the LMS algorithm along with the filter output, $y_{aug,k}$

$$\begin{aligned} W_{k+1} &= W_k + 2\mu\epsilon_k X_k \\ &= W_k + 2\mu(d_k - y_{aug,k})X_k \\ &= W_k + 2\mu(q_{d,k} + m_{d,k} + n_{d,k} - y_{aug,k})X_k \\ &= W_k + 2\mu(q_{d,k} + n_{d,k} + m_{d,k} - (X_k^T W_k + w_{2b,k}))X_k \\ &= W_k + 2\mu((q_{d,k} + n_{d,k})X_k - X_k^T W_k X_k) \end{aligned} \tag{D.1}$$

The last line was obtained by assuming that the second bias weight has ideally estimated the mean component of the desired signal ($m_{d,k} = w_{2b,k}$) and removed it from the update equation. Therefor, in using the signal model presented in Chapter 2, $d_k = q_{d,k} + r_{d,k}$ and is zero mean (13:86). This also applies to the input signal as the bias weight in the first stage has removed the mean component of the input signal. Now as k approaches infinity, the filter is assumed to reach a point in time where $E[W_{k+1} - W_k]$ equals the zero vector. Subtracting W_k from both sides and applying the expected value operator to Equation D.1 results in the following:

$$E[W_{k+1} - W_k] = 2\mu E[(q_{d,k} + n_{d,k})X_k] - E[X_k^T W_k X_k]$$

$$
\begin{aligned}
0 &= E[(q_{d,k} + n_{d,k})(Q_k + N_k)] - E[(Q_k + N_k)(Q_k^T + N_k^T)]W_k \\
0 &= E[q_{d,k}Q_k] + E[n_{d,k}N_k] - (E[Q_kQ_k^T] + E[N_kN_k^T])W_k \\
0 &= P_{q,k} + P_{n,k} - (R_q + R_n)_k W_k \qquad \text{(D.2)}
\end{aligned}
$$

The cross product terms are zero because the signal components are assumed uncorrelated and zero mean. The second line was obtain by noting that $X_k^T W_k$ is a scalar, therefor, $(X_k^T W_k)X_k = X_k(X_k^T W_k)$. In addition, the W_k was removed from the expected value assuming the filter has settled to a solution and the weight vector is not changing. Solving Equation D.2 for the weight vector yields:

$$
\begin{aligned}
W_k^{\#} &= (R_q + R_n)_k^{-1}(P_{q,k} + P_{n,k}) \\
&= (R_q + R_n)_k^{-1}P_{q,k} + (R_q + R_n)_k^{-1}P_{n,k} \\
&= W_k^* + W_k^B \qquad \text{(D.3)}
\end{aligned}
$$

Williams states that " ... any correlation between the noise components biases the weight solution away from the desired solution, W_k^*, by the amount W_k^B " (13:85). He further points out that for the single sensor case where the input signal is also the desired signal, the minimum MSE is achieved when the zero tap is one and all the other taps are zero (13:85). The filter then acts like a straight wire as was shown in Appendix D. Therefor, the LMS algorithm must be modified to prevent the filter from just passing the input signal along with the noise.

The mPa is obtained by noting that in a single sensor configuration the desired noise component is also the input noise component. Therefor, if an instantaneous estimation of the noise, $\hat{n}_{d,k}$, was known, the estimate could be subtracted from the actual noise component in the filter update. This estimate is incorporated into Equation D.1 and reduced as follows:

$$
W_{k+1} = W_k + 2\mu[(q_{d,k} + n_{d,k} - \hat{n}_{d,k})X_k - X_k^T W_k X_k]
$$

$$
\begin{aligned}
&= W_k + 2\mu(q_{d,k} + n_{d,k} - X_k^T W_k)X_k - 2\mu\hat{n}_{d,k}X_k \\
&= W_k + 2\mu\epsilon_k X_k - 2\mu\hat{n}_{d,k}X_k \qquad \text{(D.4)}
\end{aligned}
$$

For the ideal case where the estimate equals the actual noise component, $n_{d,k} = \hat{n}_{d,k}$, the noise term present at the start of Equation D.2 will go to zero and the bias term in Equation D.3 will then go to zero. This further means that the input signal and desired signal are perfectly correlated as shown below:

$$
\begin{aligned}
d_k &= q_k + n_k - n_k \\
&= q_k \qquad \text{(D.5)}
\end{aligned}
$$

Williams points out that the filter must have perfect knowledge of the instantaneous value of the noise component at each point in time. It is very unlikely that an instantaneous knowledge is achievable (13:86). However, an estimate of the average value might be acceptable. For the mPa, the average knowledge is $E[n_{d,k}X_k]$ and the estimate is then $E[\hat{n}_{d,k}X_k]$. Continuing with Equation D.4 and substituting in the estimate of the average produces the following:

$$
\begin{aligned}
W_{k+1} &= W_k + 2\mu\epsilon_k X_k - 2\mu\hat{n}_{d,k}X_k \\
&= W_k + 2\mu\epsilon_k X_k - 2\mu E[\hat{n}_{d,k}X_k] \\
&= W_k + 2\mu\epsilon_k X_k - 2\mu\hat{P}_{n,k} \qquad \text{(D.6)}
\end{aligned}
$$

$\hat{P}_{n,k}$ is the cross correlation noise vector which is dependent on time as seen by the k index and contains an estimate of the noise statistics. If the noise is assumed to be stationary, the k index can be dropped and the mPa algorithm is then

$$
W_{k+1} = W_k + 2\mu\epsilon_k X_k - 2\mu\hat{P}_n \qquad \text{(D.7)}
$$

Equation E.7 is the algorithm implemented in this thesis. The statistics for the P_n

vector are estimated from pre-stimulus noise which is detected and recorded prior to the application of the stimulus (13:89). Using the assumption that the noise is stationary, the P_n vector is not a function of time and is identical for all 50 time-sequenced filters in the TSAF.

Appendix E. *Computer Generated Data Files*

E.1 Introduction

The purpose of this appendix is to show how the computer generated data files were created. The routines to generate the files were implemented in Turbo Pascal. The files discussed here are the following:

- $AWGN_0$
- $AWGN_1$
- SIM.EEG
- SIM.MEG
- NOISE.MPA
- NOISE2.PRN

E.2 Additive White Gaussian Noise.

Figure E.1 is a block diagram of the AWGN generator used to create $AWGN_0$. The routine generates a signal number according to the algorithm shown. N is the number of times the random number generator is called. The summing routine collects N samples and outputs a single number. The entire process is repeated for each point in the data ensemble. The $AWGN_0$ file contained 100 data vectors with 50 sample points in each vector. Therefor, the AWGN generator was called 5000 times. The resulting data file was zero mean and unity variance. In addition, the data file is uncorrelated which is a property of AWGN.

The $AWGN_1$ file was generated from $AWGN_0$ as defined in Equation E.1. The routine essentially flips the data within the vectors and then reorders the vectors in the ensemble. An alternate approach is to use a different seed for the AWGN

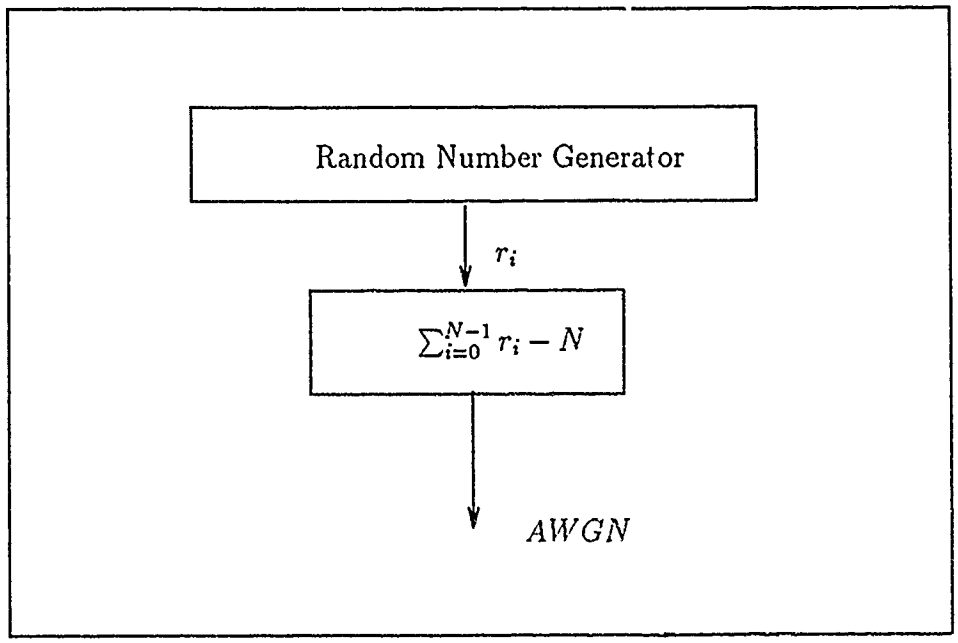

Figure E.1. Additive White Gaussian Noise Generator: The output is a single value. Repeated calls to this routine will generate an AWGN signal with zero mean and unity variance.

generator. However, Equation E.1 proved satisfactory.

$$AWGN_{1,j,k} = AWGN_{0,j-(M-1),k-(N-1)} \tag{E.1}$$

The result was that $AWGN_1$ was zero mean and unity variance and uncorrelated with $AWGN_0$.

E.3 Simulated EEG and MEG Noise

The noise files in this section were created with correlation and cross correlation. Figure E.2 is the block diagram of the routine which generated the SIM.EEG and SIM.MEG noise files. Each file contained 100 vectors with 70 points in each vector. The first feedback loop spreads correlation within the incoming AWGN. The second feedback loop then creates the cross correlation between the two files. The 0.8 value spread correlation within the file to simulate the spreading observed in the autocorrelation of the human MEG pre-stimulus noise from Chapter 3.

E.4 NOISE.MPA and NOISE2.PRN

The NOISE.MPA file was the same file as SIM.EEG. The name was different so as not to confuse the reader. The important characteristic of the NOISE.MPA is the correlation within the data file which was generated to stress the P_n estimator. The last noise file NOISE2.PRN was created with high variance as shown in Figure E.3. The multiplier was used to increase the power in the signal resulting in a lower SNR when added to the simulated EF.

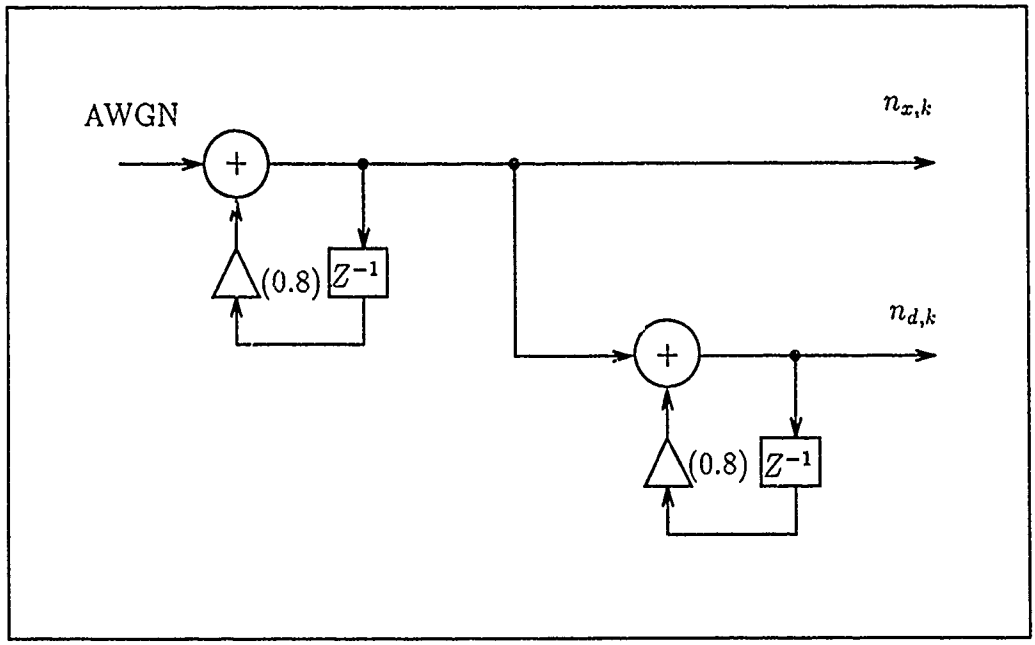

Figure E.2. Create Correlated Noise Components

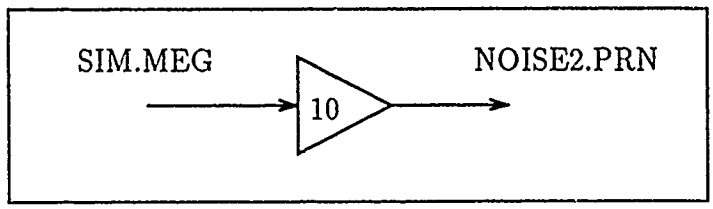

Figure E.3. Create High Variance Noise

Appendix F. *Program Listing and Description*

F.1 Introduction.

This appendix contains the program listing for the TSAF and mPa filters. The code is in Pascal and was created, edited, and debugged using Borland Turbo Pascal 5.0. An effort was made to design a modular program to facilitate debugging and enhance the readability. Variables and procedures were labeled with sufficient size to describe the variable and the function of the procedure. With this in mind, the following sections provide a brief overview of the main program loop and selected procedures.

F.2 Main Program.

The main program is a loop which is repeated based on the number of data vectors and the number of runs selected. The first procedure, clear_arrays_vectors, clears all the data arrays and vectors as Pascal does not initialize these when they are declared. The next procedure, initialize_variables, initializes all the filter parameters which are variables within the program. A "nice to have" would be a procedure to prompt the user to input the variables from the keyboard.

Next, the driver opens the necessary disk files for writing and reading and then proceeds to load data into the data vectors. This program is designed to process data in vectors while several disk files remain open during the program execution. The input files for the signal, X_j, and desired signal, d_j, are open during the entire program and are only closed at termination or to at the start of a new run. The idea behind this is that using vectors frees up a great deal of memory. In addition, the array size is limited in Pascal which means a heap pointer is required for larger arrays. To simplify the program, data is read in a vector at a time while the input files remain open. This allows an unlimited number of vectors to be processed and lets Pascal do the bookkeeping for the data pointer.

With data loaded, the program enters a nested loop to update each of the TSAF filters. This loop updates the output, error and gain vectors along with the weight array. Once through, the execution begins with the loading of the next data vector and the loop repeats. This is a simplified overview and the interested reader is invited to peruse the program listing for additional details.

F.3 General Information.

This section presents information to needed to understand the code. First, all the loops indexes are zero based which means they start at 0 and terminate at $N-1$ where N is the number of desired iterations. Second, the single letter integers are used as global loop pointers. This means that a procedure uses the variable without saving or restoring the values. Third, the three flags used in the program are *mpa_flag*, *single_sensor_flag*, and *causal_flag*. An integer 1 means the flag is set and the option selected. As an example, if *causal_flag* = 1, then the filter is causal, otherwise, the filter is non-causal. Finally, there are no nested procedure calls which means only the main program calls a procedure. This is intended to enhance the readability of the program.

F.4 Program Listing.

This is the Turbo Pascal 5.0 listing of the TSAF and mPa program.

```
Program Time_Squenced_Adaptive_Filter;
Uses CRT, Printer;

Var
   x_input_vector,
   d_input_vector,
   y_output_vector,
   bias_weight_vector,
   bias_d_weight_vector,
   e_error_vector,
   pre_x_input_vector,
   pre_d_input_vector,
```

```
    p_vector,
    gain_mu_vector      : Array[0..80] of Real;

    avg_weight_array,
    weight_array        : Array[0..80,0..20] of Real;

    filter_select,
    filter_size,
    filter_start,
    delay,
    number_data_vectors,
    number_of_runs,
    run_counter,
    mpa_flag,
    single_sensor_flag,
    causal_flag,
    post_data_size,
    pre_data_size,
    data_pointer,
    i,j,k               : Integer;

    sum_error,
    leakage,
    gain_mu,
    gain_mu_error,
    mis_adjustment,
    new_gain_mu,
    mu_update_gain,
    p_error,
    temp_var_x,
    temp_var_y                : Real;

    temp_str,
    x_input_filename,
    d_input_filename,
    w_output_filename,
    e_output_filename,
    y_output_filename    :String;

    x_input_file,
    d_input_file,
    w_output_file,
    e_output_file,
    y_output_file       : Text;

{-------------------------------------------------------------------------}
Procedure initialize_variables;

Begin
     x_input_filename       := 'd:sim_ef.prn';
     d_input_filename       := 'c:\tp\sig_files\sim_ef10.prn';
```

```
     w_output_filename      := 'd:wout.prn';
     e_output_filename      := 'd:eout.prn';
     v_output_filename      := 'd:yout.prn';

     post_data_size         := 50;
     pre_data_size          := 20;
     number_data_vectors    := 100;
     filter_size            := 15;
     number_of_runs         := 5;
     filter_start           := 20;
     leakage                := 1.0;
     mis_adjustment         := 0.15;
     mu_update_gain         := 1.0;
     causal_flag            := 0;
     mpa_flag               := 0;
     single_sensor_flag     := 0;

     sum_error              := 0;

     Textbackground(blue);
     Textcolor(yellow);
End;

{-------------------------------------------------------------------------}

Procedure open_input_disk_files;

Begin
     Assign(x_input_file,x_input_filename);
     Reset(x_input_file);

     If    single_sensor_flag = 0
     Then
         Begin
              Assign(d_input_file,d_input_filename);
              Reset(d_input_file);
         End;

End;

{-------------------------------------------------------------------------}

Procedure open_output_disk_files;

Begin

     Assign(w_output_file,w_output_filename);
     Rewrite(w_output_file);

     Assign(e_output_file,e_output_filename);
     Rewrite(e_output_file);
```

```
     Assign(y_output_file,y_output_filename);
     Rewrite(y_output_file);

End;

{-------------------------------------------------------------------------}

Procedure write_y_vector_to_disk;

{Purpose:  Write selected vector to disk.  This routine assumes the file
           is open for output and appends the vector to the text file.}

Begin

     For i:= 0 to (post_data_size - 1) Do
     Begin
          Writeln(y_output_file , i, ' ', y_output_vector[i]);
     End;

End;

{-------------------------------------------------------------------------}
Procedure write_weight_array_to_disk;

Label
     exit_write_to_disk;

Begin
     If   run_counter < (number_of_runs - 1)
     Then goto exit_write_to_disk;

     For j := 0 to (post_data_size - 1) Do
     Begin
     For i := 0 to (filter_size - 1) Do
         writeln(w_output_file,weight_array[j,i]:0:5);
     End;

     exit_write_to_disk:

End;

{-------------------------------------------------------------------------}
Procedure reset_disk_files;

{Purpose:  Moves the file pointer to the start of the file.  All subsequent
           reads start at the beginning of the file.}

Begin
     Reset(x_input_file);
```

```
     If    single_sensor_flag = 0
     Then  Reset(d_input_file);
End;

{-------------------------------------------------------------------------}
Procedure print_avg_weight_array;

Begin
     For i := 0 to (post_data_size - 1) Do
     Begin
          write(Lst,i,' ');
     For j := 0 to (filter_size - 1) Do
     Begin
          write(Lst,avg_weight_array[i,j]:7:4,' ');
     End;
         Writeln(Lst);
     End;

     Writeln(Lst,Chr(12));
End;

{-------------------------------------------------------------------------}

Procedure close_input_disk_files;

Begin
     Close(x_input_file);

     If   single_sensor_flag = 0
     Then Close(d_input_file);
End;

{-------------------------------------------------------------------------}

Procedure close_output_disk_files;

Begin
      Close(w_output_file);
      Close(e_output_file);
      Close(y_output_file);
End;

{-------------------------------------------------------------------------}

Procedure load_input_vectors;


Begin
     For i := 0 to (post_data_size - 1) Do
     Begin
          ReadLn(x_input_file , temp_var_x, x_input_vector[i]);
```

```
          If    single_sensor_flag = 0
          Then  ReadLn(d_input_file ,temp_var_x, d_input_vector[i])
          Else  d_input_vector[i] := x_input_vector[i];
     End;

End;

{-------------------------------------------------------------------------}

Procedure load_pre_input_vectors;

{Purpose:  Load the pre-stimulus data into the appropriate vectors.  If the
           data files do not have pre-stimulus data, then pre_data_size should
           be set to 0 !!!.}

Begin
    For i := 0 to (pre_data_size - 1) Do
    Begin
        ReadLn(x_input_file, temp_var_x, pre_x_input_vector[i]);

        If    single_sensor_flag = 0
        Then  ReadLn(d_input_file, temp_var_x, pre_d_input_vector[i])
        Else  pre_d_input_vector[i] := pre_x_input_vector[i];
     End;

End;

{-------------------------------------------------------------------------}

Procedure test_filter_size;

{Purpose: to make filter_size odd and set DELAY.  Making the filter size odd
          simplifies the algorithms for a non-causal filter because the filter
          is symetrical about the center tap.  Also,
          DELAY is set to filter_size if causal, otherwise it is the number
          of taps to either side of the center tap.}

Begin
     temp_var_x := Frac(filter_size/2);

     If   temp_var_x = 0
     Then filter_size := filter_size + 1;

     If   causal_flag = 1
     Then delay :=  filter_size - 1
     Else delay := (filter_size - 1) Div 2;

     End;

{-------------------------------------------------------------------------}
```

```
Procedure update_screen;

Begin
     If data_pointer = 0 Then
     Begin
        ClrScr;
        GotoXY(1,2);
        Writeln('   Filter Size: ',filter_size);
        Writeln('Number of Runs: ',number_of_runs);
        Writeln('   Causal Flag: ',causal_flag);
        Writeln('      mPa Flag: ',mpa_flag);
        Writeln(' Misadjustment: ',mis_adjustment:0:3);

     End;

     GotoXY(1,8);
     Write('      Working on Run #: ','<',run_counter + 1,'>    ');
     GotoXY(1,10);
     Write('Processing Data Vector: ','<',data_pointer,'>  ');
     GotoXY(1,12);
     Write('        Variance Error: ','<',sum_error:0:4,'>');

End;
{-------------------------------------------------------------------------}
Procedure update_screen_2;
Begin
     Textcolor(yellow+blink);
     GotoXY(1,8);
     Write('      Filtering');
     Textcolor(yellow);
     GotoXY(1,10);
     Write('Processing Data Vector: ','<',data_pointer,'>  ');
End;

{===================== Procedures for gain_mu_vector ====================}

Procedure update_gain_mu_vector;

Begin
     gain_mu_vector[filter_select] := mu_update_gain * new_gain_mu;



End;
{-------------------------------------------------------------------------}

Procedure calculate_new_gain_mu;

{Purpose  :  To calculate the gain constant used in the weight updates.  This
             procedure scales the gain constant using the input signal energy
```

```
            and the number of filter taps.}

Var
     variance             : Real;

Label
     Skip_New_Gain_Loop;

Begin
     variance           := 0;

     For i := 0 to (filter_size - 1) Do
     Begin
          If   ( filter_select - delay + i < 0 )
               or
               ( filter_select - delay + i > ( post_data_size - 1 ))
          Then
               GoTo Skip_New_Gain_Loop;

          variance := variance +
                      Sqr(x_input_vector[filter_select - delay + i]);


          Skip_New_Gain_Loop:
     End;

     variance := variance / filter_size;

     new_gain_mu := (mis_adjustment) /
                    (filter_size*(variance + 0.001)*(run_counter + 1));


End;


{=================== Procedure for bias_weight_update ======================}

Procedure update_bias_weight_vectors;

{Purpose:  Estimate the mean of the input signal and desired.  The weight
           of the error is decreased with each data vector.  If the filter is
           doing multiple runs, then the bias weight is not updated after the
           first run and remains fixed for all additional runs.}

Begin

     If    run_counter = 0
     Then
         For i := 0 to (post_data_size - 1) Do
         Begin
             bias_weight_vector[i] := bias_weight_vector[i] +
```

```
                                        (2.0 * 0.5 /
                                        (data_pointer + 1)) *
                                        (x_input_vector[i] -
                                          bias_weight_vector[i]);

            bias_d_weight_vector[i] := bias_d_weight_vector[i] +
                                        (2.0 * 0.5 /
                                        (data_pointer + 1)) *
                                        (d_input_vector[i]-
                                         bias_d_weight_vector[i]);
          End;
End;

{-------------------------------------------------------------------------}
Procedure remove_bias_from_input_vectors;

Begin
     For i := 0 to (post_data_size - 1) Do
     Begin
          x_input_vector[i] := x_input_vector[i] -
                               bias_weight_vector[i];

          d_input_vector[i] := d_input_vector[i] -
                               bias_d_weight_vector[i];
     End;
End;
{================== Procedures for filter updates ========================}

Procedure update_y_output_vector;

Var
   sum_var : Real;

Label end_loop;

Begin
     sum_var := 0;

     For i:= 0 to (filter_size -1) Do
     Begin
          If   (filter_select - delay + i < 0)
               or
               (filter_select - delay + i > (post_data_size - 1))

          Then Goto End_Loop;

          sum_var := sum_var +
                     weight_array[filter_select, i] *
                     x_input_vector[filter_select - delay + i];

          End_Loop:
```

```
     End;

     y_output_vector[filter_select] := sum_var;

End;

{-------------------------------------------------------------------------}
Procedure add_bias_to_y_vector;

Begin
     For i := 0 to (post_data_size -1) Do
     y_output_vector[i] := y_output_vector[i] +
                           bias_d_weight_vector[i];
End;

{-------------------------------------------------------------------------}

Procedure update_e_error_vector;

Begin
     e_error_vector[filter_select] := d_input_vector[filter_select] -
                                      y_output_vector[filter_select];
End;

{-------------------------------------------------------------------------}
Procedure sum_error_vector;

Begin
     sum_error := 0;

     For i:= 0 to (post_data_size - 1) Do
     sum_error := sum_error + Sqr(e_error_vector[i]) / post_data_size;

     Writeln(e_output_file, sum_error);

End;
{-------------------------------------------------------------------------}

Procedure update_weight_array;

Label Skip_Weight_Update;

Begin

     For i := 0 to (filter_size - 1) Do
     Begin

         If   (filter_select - delay + i < 0)
              or
              (filter_select - delay + i > (post_data_size -1))
```

```
        Then Goto Skip_Weight_Update;

        weight_array[filter_select,i] := leakage * weight_array[filter_select,i] +
                                         2 * gain_mu_vector[filter_select] *
                                         e_error_vector[filter_select] *
                                         x_input_vector[filter_select - delay + i];

        If    mpa_flag = 1
        Then
        weight_array[filter_select,i] := weight_array[filter_select,i] -
                                         2 * gain_mu_vector[filter_select] *
                                         p_vector[i];


        Skip_Weight_Update:

        End;

End;

{-------------------------------------------------------------------------}
Procedure clear_arrays_vectors;

Begin
     For i := 0 to 79 Do
     Begin
        x_input_vector[i]         := 0;
        d_input_vector[i]         := 0;
        y_output_vector[i]        := 0;
        e_error_vector[i]         := 0;
        gain_mu_vector[i]         := 0;
        pre_x_input_vector[i]     := 0;
        pre_d_input_vector[i]     := 0;
        p_vector[i]               := 0;
        bias_weight_vector[i]     := 0;
        bias_d_weight_vector[i]   := 0;
      End;

      For i := 0 to 79 Do
      Begin
           For j := 0 to 20 Do
           Begin
                avg_weight_array[i,j] := 0;
                weight_array[i,j]     := 0;
           End;
       End;

End;
{-------------------------------------------------------------------------}

Procedure update_p_vector;
```

```
Var
   time_index,
   tap           : Integer;


Label Skip_pre_vector_update,
      mpa_not_selected;

Begin
     If   (mpa_flag = 0)
          or
          (run_counter > 0)

     Then Goto mpa_not_selected;

     For i := 0 to (pre_data_size - 1) Do
     Begin
          For tap := 0 to (filter_size - 1) Do
          Begin

               If   (tap - delay + i < 0)
                    or
                    (tap - delay + i > pre_data_size - 1)

               Then Goto Skip_pre_vector_update;


               p_vector[tap] := p_vector[tap] +
                                (pre_x_input_vector[tap - delay + i] *
                                pre_d_input_vector[i]) /
                                (number_data_vectors * pre_data_size);


          Skip_pre_vector_update:
          End;
      End;

      mpa_not_selected:
End;

{-------------------------------------------------------------------------}
Procedure freeze_weights;

Begin
     For i := 0 to (post_data_size - 1) Do
     Begin
          For j := 0 to (filter_size - 1) Do
          weight_array[i,j] := avg_weight_array[i,j];
     End;
End;
```

```
{-------------------------------------------------------------------------}
Procedure do_avg_weight_array;

Label
     exit_avg;

Begin
     If   run_counter < (number_of_runs - 1)
     Then goto exit_avg;

     if data_pointer = 1
     Then Begin
               textcolor(yellow+blink);
               GotoXY(40,2);
               writeln('Averaging Weights');
               textcolor(yellow);
          End;

     For i := 0 to (post_data_size - 1) Do
     Begin
          For j := 0 to (filter_size - 1) Do

          avg_weight_array[i,j] := avg_weight_array[i,j] +
                                   (2.0)*(0.5)/(data_pointer + 1) *
                                   (weight_array[i,j] -
                                   avg_weight_array[i,j]);
     End;

     Exit_avg:
End;

{-------------------------------------------------------------------------}
Procedure reset_filter_start;

{Purpose:  filter_start will turn the filter on after the data_pointer has
           exceed the set value of filter_start.  This allows the bias weight
           vector to get a jump start on the input data before the filter.  This
           reduces the noise passed to the filter from the bias weight.  After
           the filter is started, this routine clears filter_start so the filter
           is on continuously.}

Begin
     filter_start := 0;
End;

{<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<}
{<<<<<<<<<<<<<<<<<<<<<<<      Main_Program      >>>>>>>>>>>>>>>>>>>>>>>>>>>>}

Begin
     ClrScr;
```

```
initialize_variables;
clear_arrays_vectors;
test_filter_size;
open_input_disk_files;
open_output_disk_files;

For run_counter := 0 to (number_of_runs - 1) Do
Begin
     reset_disk_files;

     For data_pointer := 0 to (number_data_vectors - 1) Do
     Begin
          load_pre_input_vectors;
          load_input_vectors;

          update_screen;
          update_p_vector;
          update_bias_weight_vectors;
          remove_bias_from_input_vectors;

          If   data_pointer >= filter_start
          Then
               For filter_select := 0 to (post_data_size - 1) Do
               Begin
                    update_y_output_vector;
                    update_e_error_vector;
                    calculate_new_gain_mu;
                    update_gain_mu_vector;
                    update_weight_array;
                    reset_filter_start;
               End;

               do_avg_weight_array;
               sum_error_vector;

          End;


 End;

 {====================Freeze Weights and Filter===============}

 freeze_weights;
 reset_disk_files;
 ClrScr;

 For data_pointer := 0 to (number_data_vectors - 1) Do
 Begin
      load_pre_input_vectors;
      load_input_vectors;
```

```
            remove_bias_    .input_vectors;

            For filter_selec. := 0 to (post_data_size - 1) Do
            Begin
                 update_y_output_vector;
            End;

            update_screen_2;
            add_bias_to_y_vector;
            write_y_vector_to_disk;

       End;

       write_weight_array_to_disk;
       {print_avg_weight_array;}

       close_input_disk_files;
       close_output_disk_files;

End.
```

Bibliography

1. Davenport, Wilbur B., Jr. *Probability and Random Processes.* McGraw-Hill Book Company, Inc., New York, 1970.
2. Ferrara, Earl Roy, Jr. *The Time Sequenced Adaptive Filter.* PhD dissertation. Stanford University, Stanford CA, 1978.
3. Ferrara, Earl Roy, Jr. and Bernard Widrow. "The Time-Sequenced Adaptive Filter," *IEEE Transactions on Circuits and Systems, 28*: 519-523 (June 1981).
4. Ludeman, Lonnie C. *Fundamentals of Digital Signal Processing* Harper & Row, Publishers, Inc., New York, 1986.
5. Oppenheim, Alan V. and others. *Signals and Systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.
6. Schmidt, J.,William and Robert P. Davis. *Foundations of Analysis in Operations Research.* Academic Press, Inc., Orlando, Florida, 1981.
7. Shanmugan, K. Sam and Arthur M. Breipohl. *Random Signals: Detection, Estimation and Analysis.* John Wiley & Sons, Inc. New York, 1988.
8. Weinberg, Harold and others. *Biomagnetism: Applications & Theory.* Pergamon Press Inc., Elmsford, NY, 1985.
9. Westerkamp, John J. and Jorge I. Aunon. "Optimum Multielectrode A Posteri ·ri Estimation of Single-Response Evoked Potentials." *IEEE Transactions on Biomedical Engineering, Vol. BME-34, No. 1*: 13-22 (January 1987).
10. Widrow, Bernard and Samuel D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.
11. Widrow, Bernard and others. "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter." *Proceedings of the IEEE, Vol 64, No. 8*: 1151-1162 (August 1976).
12. Williams, Robert. Midterm Project in EENG-699, Adaptive Filters. School of Engineering, Air Force Institute of Technology (AU), Wright- Patterson AFB OH, February 1990.
13. Williams, Robert. *Adaptive Filtering of Nonstationary Signals Using a Modified P-Vector Algorithm.* PhD dissertation. University of Dayton, Dayton, Ohio, December 1989.
14. Williamson, Samuel J. and others. *Biomagnetism: An Interdisciplinary Approach.* Plenum Press, New York, NY, 1983.
15. Wood, Capt Roger A. *Electro-Encephalogram Based Adaptiv Estimation of Magneto-Encephalogram Signals.* MS Thesis AFIT/GE/EN/88D. School of

Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1988 (AD-A202662).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1990	*Master's Thesis*

4. TITLE AND SUBTITLE

Estimation of Evoked Fields Using a Time-Sequenced Adaptive Filter With the Modified P-Vector Algorithm

5. FUNDING NUMBERS

6. AUTHOR(S)

Jeffery Allen Kepley, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/90D-29

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A time sequenced adaptive filter is developed to estimate visually evoked fields (EF) using visually evoked potentials (EP). These non-stationary signals are buried in strong background noise. The two types of noise are magnetoencephalogram (MEG) and electroencephalogram (EEG). The filter implementation is based on the Ferrara Time Sequenced Adaptive Filter (TSAF) using the Least-Mean-Square (LMS) algorithm and the Williams modified P-vector algorithm (mPa). This essentially results in two filters, the $TSAF_{LMS}$ and the $TSAF_{mPa}$ respectively. A two stage filter structure is used in which the first stage removes the time-varying mean of the input signals allowing the second stage to process zero-mean signals. The theory for the $TSAF_{LMS}$ and $TSAF_{mPa}$ filters is overviewed with the input signals to the filters modelled as the sum of three uncorrelated components: average signal response, signal jitter, and noise. Using simulated EF and EP signals buried in correlated noise, the $TSAF_{mPa}$ is shown to out perform the $TSAF_{LMS}$ for the specific case presented. The $TSAF_{mPa}$ is unique in that the filter uses an estimate of the cross correlation statistics of the pre-stimulus noise in the filter update equation. The noise statistics, contained in the P_n vector, are assumed to be time-invariant which is a fundamental assumption used in this research.

14. SUBJECT TERMS

Evoked Field, Evoked Potential, Adaptive Filter, Time Sequenced Adaptive Filter, Modified P-Vector Algorithm

15. NUMBER OF PAGES

180

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1 Agency Use Only *(Leave blank)*

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es) Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as Prepared in cooperation with..., Trans. of..., To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- DOE - See authorities.
- NASA - See Handbook NHB 2200.2.
- NTIS - Leave blank.

Block 12b. Distribution Code.

- DOD - Leave blank.
- DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- NASA - Leave blank.
- NTIS - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20 Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.